Dentrix® API Developer Guide

Version 2.1

1220 South 630 East, Suite 100
American Fork, Utah 84003
www.dentrix.com/ddp
ddp@dentrix.com

Publication Date

May 2015

Copyright

Software License Notice

Licenses and Trademarks

Table of Contents

Revision History

Version	Date	Change
1.1	04/13/2011	*Initial release*
1.2	09/12/2011	• *New Stored Procedure functions were added to retrieve data for Insurance Plans, Patient's Medical Alerts, Document Types, Provider Class and Color, Appointment Status Codes, Office Journal Entries, Referral Sources and Referral Specialty* • ***v_patient_recall*** *Table View was deprecated and replaced with* ***sp_getallpatientrecalls*** *Stored Procedure* • ***v_provider*** *Table View was deprecated and replaced with* ***sp_getallproviders*** *Stored Procedure*
1.3	11/14/2011	• *New Stored Procedure function was added to retrieve data for the Appointment Book Day Note*
1.4	11/30/2011	• *Patient GUID fields and/or inputs were added to Tables Views and Stored Procedures where applicable*
1.4.2	2/13/2013	• ***v_patient*** *Table View was deprecated and replaced with* ***sp_patient*** *Stored Procedure* • *The following changes were added for Create User utility:* o *Security-related changes* o *Ability to update the Dentrix API with the latest functionality* o *Removed requirement to run Create User utility with a parameter* • ***status_id*** *column was added to the* ***v_appointment*** *Table View*
2.0	10/9/2013	• ***sp_getpatientmedicalalerts*** *stored procedure was deprecated and replaced with* ***sp_getpatientmedalerts*** • *New Table Views were added retrieve data for Adjustment Types, Payment Types, Billing Types, Practice Medical Alerts, Practice information, Patient's Insurance Information, Patient Clinical Notes, Employers, Appointment Book Events, Appointment Types, Appointment Check List, Initial Appointment Reasons* • *New Stored Procedures were added to retrieve data for Patient's Recall, Patient's Appointment Information* • *Additional columns were added to the* ***v_appointment*** *Table View* • *Updated several table views and stored procedures to retrieve data more quickly than in previous versions*

		• *Made miscellaneous improvements to the Create User utility and also removed the API Update capability from it* • *Added more sample code*
2.1	May 2015	• *New Table Views / Stored Procedures were added retrieve data for Account Notes, Appointment Amount Setting, Dental Lab Cases, Insurance Plan Notes, Insured Subscribers, Labs, Operatory Default Schedule, Operatory Exceptions, Patient Address, Perio Exams, Practice Default Schedule, Practice Schedule Exceptions, Procedure Codes, Provider Default Schedule, Provider Schedule Exceptions, Provider Time Blocks, Recall Types, Referral Specialties and Transactions (Treatment Planned Procedures, Completed Procedures, Adjustments and Payments).* • *Made miscellaneous improvements to the Create User utility* • *Added details for connecting to the API from a Java app* • *Added details for opening a Dentrix module (from your app) with a patient selected* • *Deprecated sp_getallreferralspecialties stored procedure* • *Deprecated v_appointment view*

Getting Started

Welcome to the Dentrix Developer Program. Participation in this program provides you with access to the Dentrix API as well as other tools to assist you in developing and delivering integrated software solutions to Dentrix users. Follow this guide to get started, but remember to consult our additional online resources to help you make the most of this program.

To begin using the Table Views and Stored Procedures provided in the Dentrix API, you will need the following:

- A user name and password provided by Henry Schein Practice Solutions to grant you access to the table views and stored procedures
- Dentrix G5 or above installed on at least one computer
- Dentrix Software Development Kit (*DentrixSDK_2.1.zip*), which includes the following:
    - This Dentrix API Developer Guide document
    - *Table View Tutorial* Microsoft® Visual Studio project and other sample code
    - Create User utility (see the *Requesting Access to the Dentrix API* section below for instructions). *NOTE: If you have completed the Dentrix Developer Program registration process and paid the applicable registration fee, but have not yet received your Create User utility, contact the Dentrix Developer Program team at DDP@Dentrix.com.*

Online Resources

You can download the Dentrix G5 and higher software, SDK, sample code and other developer tools by logging into the Dentrix Developer Program website at ddp.dentrix.com and then browsing to the Resource Center > Downloads page. You will find the current sample code and file in the *DentrixSDK_2.1.zip* file.

For complete system recommendations and requirements for running the Dentrix G5 and above software, the current System Requirements document can be obtained from the Dentrix website at www.dentrix.com. You can find answers to questions about the general use of Dentrix by referring to the user's guide or by using the context-sensitive help available in the Help menu of each Dentrix module. The user guide for Dentrix is installed with Dentrix G5. If you are unable above to find an answer to your questions about the Dentrix software using these methods, contact Dentrix Software Support at 1-800-DENTRIX.

When contacting Dentrix Software Support, be prepared to give the following information:

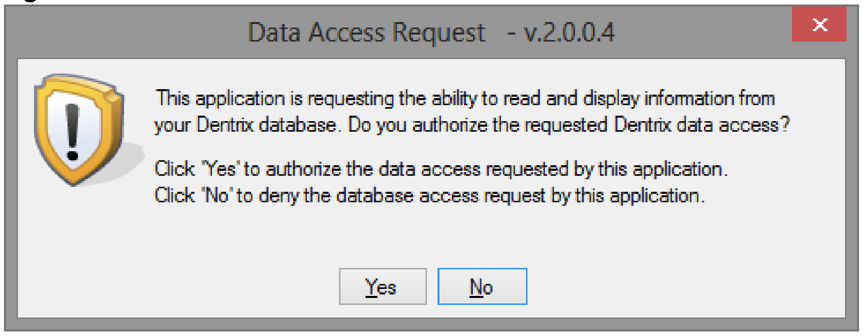
- Your name and the name of your organization
- The Dentrix customer number assigned to your organization
- The version number of Dentrix being used

Requesting Access to Dentrix API

In order to access the API Table Views and Stored Procedures detailed in this document, access must be requested using the Create User utility. The Create User utility (named ***CreateUser_<Your API Username>_<Version>.exe***) adds your API user account a Dentrix G5 or higher database and gives you access to the Table Views and Stored Procedures. To run the Create User utility, do the following:

1. Copy the Create User utility (self-extracting zip file) to any location on any computer running Dentrix G5 or higher. The Create User application does not need to be run on every workstation on a Dentrix network, but does need to be run on every Dentrix G5 or higher database that you will be accessing through the Dentrix API.
2. When the Create User utility is successfully initialized, the user will be prompted to authorize Dentrix data access for your application as illustrated in Figure 1. If passwords have been enabled in Dentrix for the practice, the user will also be required to enter their Dentrix password (and will specifically need rights to the "Password Administration" security option in Dentrix) in order to authorize data access to your application.

Figure 1

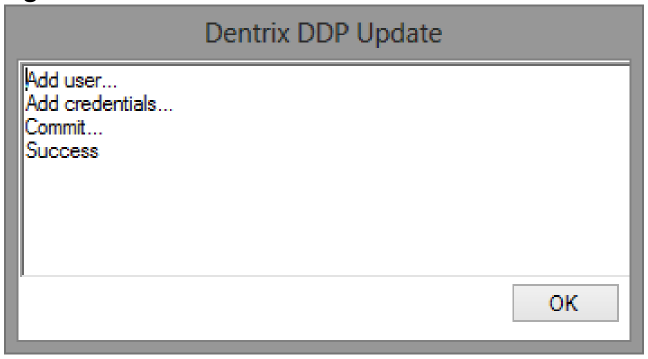


This application is requesting the ability to read and display information from your Dentrix database. Do you authorize the requested Dentrix data access?

Click 'Yes' to authorize the data access requested by this application.
Click 'No' to deny the database access request by this application.

3. Once the user has authorized data access for your application from the Data Access Request prompt, the Create User utility will do the following:
   - Add your API user account and credentials to the Dentrix G5 or higher database
   - Display the progress of the update (including any errors that may occur) in a *Dentrix DDP Update* window as shown in Figure 2
   - Once your API user account and credentials have been successfully added, a "Success" status will be displayed and the OK button on the *Dentrix DDP Update* window will become enabled to close the window

Figure 2



```
Add user...
Add credentials...
Commit...
Success
```

4. In addition to displaying the Create User progress in the *Dentrix DDP Update* window, the progress of the update will also be written to a CreateUser.log file in the following directory under the Common path on the Dentrix server: \Doc\Exports\Installs. The Common path for Dentrix is set in the registry as follows:

32-bit OS

```
HKEY_LOCAL_MACHINE\SOFTWARE\Dentrix Dental Systems, Inc.\Dentrix\General\CommonPath
```

64-bit OS

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Dentrix Dental Systems,
Inc.\Dentrix\General\CommonPath
```

NOTE: It is possible that the message "Could not grant all rights" will appear in the Dentrix DDP Update window in certain situations when the Create User utility is run. This message indicates that one or more API views or stored procedures that the Create User utility is attempting to grant the user access to do not exist in the installed version of Dentrix. To determine specifically which views and stored procedures the Create User utility could not grant the user access to, check the most recent entry (bottom) of the CreateUser.log stored in the \Doc\Exports\Installs directory on the Dentrix server (see above for location). To see the version of Dentrix in which each API view or stored procedure is available, please see the API Table Views and Stored Procedures list in this document.

The table below lists all Dentrix API Table Views and Stored Procedures described in this document.

Table Views and Stored Procedures (List)

Table View / Stored Procedure Name	Title	Rev #	Dentrix Availability
v_account_notes	Account / Guarantor Notes	1	G6+
v_address	Patient Address	1	G6+
v_adjustment_types	Adjustment Types	1	G5.2+
v_appointment_checklist	Appointment Check List	1	G5.2+
v_appointment_types	Appointment Types	1	G5.2+
v_appointment	Appointments	2	G5+
v_appt_book_events	Appointment Book Events	1	G5.2+
v_appt_status_codes	Appointment Status Codes	1	G5+
v_billing_types	Billing Types	1	G5.2+
v_clinical_notes	Patient Clinical Notes	2	G5.2+
v_document_types	Document Types	2	G5+
v_employers	Employers	1	G5.2+
v_guarantor_balance	Guarantor’s Balance	1	G5+
v_initial_reasons	Appointment Initial Reasons	1	G5.2+
v_insurance_plans	Insurance Plans	1	G5+
v_insurance_plan_notes	Insurance Plan Notes	1	G6+
v_insured	Insured Subscribers	1	G6+
v_lab	Dental Labs	1	G6+
v_lab_case	Dental Lab Cases	1	G6+
v_operatory	Operatories	1	G5+
v_notes	Notes	1	G6+
v_patient	Patient Demographics (View)	2	G5+
v_patient_insurance	Patient’s Dental Insurance Information	1	G5.1+
v_payment_types	Payment Types	1	G5.2+
v_perio	Perio Exam Records	2	G6+
v_practice_information	Practice	1	G5.2+
v_practice_medical_alerts	Practice Medical Alerts	1	G5.2+
v_proccodes	Procedure Codes	1	G6+

v_proclog	Ledger Transactions	0	G6+
v_provider	Providers (View)	1	G5+
v_recall_type	Recall Type	1	G6+
v_referral_specialty	Referral Specialty	1	G6+
v_rx_patient	Patient Prescriptions	1	G5+
v_rxlist	Practice Prescriptions	1	G5+
v_staff	Staff	1	G5+
sp_appointments	Modified Appointments	2	G5.2+
sp_apptbookdaynote	Appointment Book Day Note	3	G5+
sp_getaccountpayments	Account Payments	2	G5+
sp_getallpatientrecalls	All Patients' Recall Information	3	G5+
sp_getallpatientsappointmentinfo	All Patients' Appointment Information	1	G5+
sp_getallpatientsinsuranceinfo	All Patients' Dental Insurance Information	2	G5+
sp_getallproviders	Providers (Stored Procedure)	2	G5+
sp_getallproviderspecialties	Provider Specialty	2	G5+
sp_getallprovidersproduction	All Providers' Production	2	G5+
sp_getapptamountsetting	Appointment Amount Setting	1	G6+
sp_getappointmentunitsize	Appointment Unit Size	2	G6+
sp_getguarpmtagreement	Guarantor's Payment Agreement	1	G5+
sp_getpatientadjustments	Patient's Adjustments	2	G5+
sp_getpatientappointmentinfo	Patient's Appointment Information	1	G5.2+
sp_getpatientbalance	Patient's Balance	2	G5+
sp_getpatientmedalerts	Patient's Medical Alerts	1	G5+
sp_getpatientnextapptdatetime	Patient's Next Appt Date and Time	3	G5+
sp_getpatientofficejournal	Patient's Office Journal Entries	2	G5+
sp_getpatientpreviousapptdatetime	Patient's Previous Appt Date and Time	3	G5+
sp_getpatientprocedures	Patient's Completed Procedures	3	G5+
sp_getpatientrecalls	Patient Recall	3	G5.2+
sp_getpatientreferrals	Patient's Referrals	3	G5+
sp_getpatienttreatmentplan	Patient's Treatment Planned Procedures	3	G5+
sp_getperioexam	Perio Exam Data	2	G6+
sp_getpracticeschedule	Practice Default Schedule	1	G6+
sp_getproviderschedule	Provider Default Schedule	1	G6+
sp_getoperatoryschedule	Operatory Default Schedule	1	G6+
sp_getpracticeschedexceptions	Practice Schedule Exceptions	1	G6+

sp_getprovschedexceptions	Provider Schedule Exceptions	1	G6+
sp_getoperatoryschedexceptions	Operatory Schedule Exceptions	1	G6+
sp_getprovidermonthtotals	Provider Analysis Totals	3	G5+
sp_getproviderofficejournal	Provider's Office Journal Entries	2	G5+
sp_getproviderproduction	Provider's Production	2	G5+
sp_getprovtimeblockdefault	Provider Time Blocks (Default)	1	G6+
sp_getprovtimeblockschedule	Provider Time Blocks (Specified Week)	1	G6+
sp_getreferralofficejournal	Referral's Office Journal Entries	1	G5+
sp_getreferralsource	Referral Source	2	G5+
sp_patient	Patient Demographics (Stored Procedure)	2	G5.1+

Dentrix G5 and Above Versions

Base Version	Title / Description	Build Number(s)	API Version	Released
G5	Original G5 Release	15.0.450.0	1.4	January 2012
G5	G5 Final Revision	15.0.455.0	1.4	April 2012
G5	G5 Final Rev. + 2012 ADA Claim Form	15.0.456.0	1.4	May 2012
G5	G5 Hotfix 2	15.1.232.0	1.4	November 2012
G5.1	G5 Productivity Pack 1	15.1.256.0	1.4.2	December 2012
G5.1	G5 Re-Master with PP1	15.1.257.0	1.4.2	February 2013
G5.1	G5.1 Hotfix 1	15.1.294.0	1.4.2	February 2013
G5.1	G5.1 Hotfix 2	15.1.312.0	1.4.2	March 2013
G5.1	G5.1 Hotfix 2.5	15.1.312.1, 15.1.312.2	1.4.2	July 2013
G5.1	G5.1 Update 3	15.2.148.0, 15.2.159.0	1.4.2	September 2013
G5.2	G5.2 Release	15.2.211.1	2.0	September 2013
G5.2.1	G5.2 Hotfix 1	15.2.229.2	2.0	September 2014
G5.2.1	G5.2 Re-Master	15.2.239.0	2.0	September 2014
G6	G6 Release	16.0.309.0	2.1	May 2015

Check out the 'Dentrix G5 and Above Version Detection Tool' (for detecting the version of Dentrix G5 or higher installed) available online. You can download this tool (in .bat format) by logging into the Dentrix Developer Program website at ddp.dentrix.com and then browsing to the Resource Center > Downloads page.

Table Views and Stored Procedures (Details)

This section details the table views and stored procedures currently available in the Dentrix API. *NOTE: For stored procedures that require a Patient GUID input, if a patient cannot be found according to the Patient GUID value passed in, it is recommended that you provide a method for users to find a possible match for the patient based on other criteria, such as patient name, phone number, etc.*

1. Appointments and Events

1.1 Appointments

View Name: v_appt
Revision: 3
Description: This view retrieves scheduled appointments.
Example: select * from admin.v_appt

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
appointment_id	Appointment ID	INTEGER	4
appointment_date	Appointment Date	DATE	4
operatory_id	Operatory ID in which the Appointment is Scheduled	CHARACTER	4
provider_id	Provider ID for the Appointment	CHARACTER	4
Broken	Flag for broken and unscheduled appointments: - False=Scheduled appointment - True=Unscheduled/broken appointment	BIT	1
patient_name	Patient Name (Last Name, First Name)	CHARACTER	32
patient_id	Patient ID. This is 0 if the appointment is for a new patient.	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
length	Appointment Length	SMALLINT	2
amount	Appointment amount	MONEY	10
broken_date	Used only for unscheduled/broken appointments. The date of the appointment when it is broken or moved to unscheduled; or if the appointment has never been scheduled, the current system	DATE	4

	date when appointment was unscheduled (wait/will call).		
appt_flag	Appointment "Schedule Type" flag: - 0=FIXED - 1=OPEN - 2=ASAP	TINYINT	1
patient_phone	Patient Phone Number	CHARACTER	17
reason	Appointment Reason	CHARACTER	61
start_hour	Appointment Start Hour	TINYINT	1
start_minute	Appointment Start Minute	TINYINT	1
status_id	Appointment Status ID: - 0=No Status - 1 through 10 = Status ID corresponding to the user-defined status specified in the *status_description* column - 100=Broken appointment - 101=Wait/Will Call (Unscheduled) appointment - -106=Completed - 150=Completed	TINYINT	1
production_type	Production Type: - 0=None - 1=General - 2=High Production - 3=Medium Production - 4=Low Production	TINYINT	1
newpatient_addressid	Address ID for new patient appointments. Links to appointment's new patient Addresses record (see *address_addrid* in v_address). This is 0 if the appointment is not for a new patient. An Address record is created whether or not an address and phone number is entered for the new patient.	INTEGER	4
auto_update_cc	Indicates whether or not Continuing Care (Recall) is automatically attached to the appointment according to the appointment reason: - 0= checkbox for "Use Reason to Auto Update CC" is not marked - 1="Use Reason to Auto Update CC" is marked	BIT	1
amountset	Indicates whether or not the scheduled amount for the appointment should always be automatically calculated.	BIT	1

	- False= the checkbox for "Set Amount" is not marked - True= the checkbox for "Set Amount" is marked There is a setting from Practice Appointment Setup that overrides this to always calculate. Only recalculates if the appointment is edited (whether or not anything is changed).		
ref_by_type	Type of referral for new patient appointments. - 0= Referred by patient - 1= Referred by Dr/other	TINYINT	1
lab_case_appt	Flag to indicate if the appointment is related to a lab case: - True= Appointment is related to a lab case. - False= Appointment is not related to a lab case	BIT	1
addtnl_provider_id	ID for Additional Provider for the Appointment	CHARACTER	4
ref_id	Referral ID for new patient appointment. Links to a Referral Source record.	INTEGER	4
followup	Bits to indicate the state of the "Appointment Check List" checkboxes for the appointment. There may be 0 to 12 checkboxes. Each bit is a Foreign Key that links to a record in the Appointment Check List table view for the appointment check list items. May be 0.	SMALLINT	2
codeid1	IDs for the appointment reason. Each ID links to Procedure Codes record (see *proccodeid* in v_proccodes) or Procedure Log record for treatment plan procedures (see *procid* in v_proclog where chartstatus = 105), according to the corresponding *codetype* column in this view. The procedures that will be completed for the patient when the appointment is "set complete". May be 0.	INTEGER	4
codeid2	See codeid1.	INTEGER	4
codeid3	See codeid1.	INTEGER	4
codeid4	See codeid1.	INTEGER	4
codeid5	See codeid1.	INTEGER	4

codeid6	See codeid1.	INTEGER	4
codeid7	See codeid1.	INTEGER	4
codeid8	See codeid1.	INTEGER	4
codeid9	See codeid1.	INTEGER	4
codeid10	See codeid1.	INTEGER	4
codeid11	See codeid1.	INTEGER	4
codeid12	See codeid1.	INTEGER	4
codeid13	See codeid1.	INTEGER	4
codeid14	See codeid1.	INTEGER	4
codeid15	See codeid1.	INTEGER	4
codeid16	See codeid1.	INTEGER	4
codeid17	See codeid1.	INTEGER	4
codeid18	See codeid1.	INTEGER	4
codeid19	See codeid1.	INTEGER	4
codeid20	See codeid1.	INTEGER	4
staff_id	Staff ID for the Appointment	CHARACTER	4
modified_date	The date the appointment was last changed.	DATE	4
codetype1	Type of CodeId#. - 0= the corresponding CodeId# links to a Procedure Code record - 1= the corresponding CodeId# links to a Procedure Log record (treatment plan procedure)	TINYINT	1
codetype2	See CodeType#.	TINYINT	1
codetype3	See CodeType#.	TINYINT	1
codetype4	See CodeType#.	TINYINT	1
codetype5	See CodeType#.	TINYINT	1
codetype6	See CodeType#.	TINYINT	1
codetype7	See CodeType#.	TINYINT	1
codetype8	See CodeType#.	TINYINT	1
codetype9	See CodeType#.	TINYINT	1
codetype10	See CodeType#.	TINYINT	1
codetype11	See CodeType#.	TINYINT	1
codetype12	See CodeType#.	TINYINT	1
codetype13	See CodeType#.	TINYINT	1
codetype14	See CodeType#.	TINYINT	1
codetype15	See CodeType#.	TINYINT	1
codetype16	See CodeType#.	TINYINT	1
codetype17	See CodeType#.	TINYINT	1
codetype18	See CodeType#.	TINYINT	1
codetype19	See CodeType#.	TINYINT	1

codetype20	See CodeType#.	TINYINT	1
create_date	Date the appointment was created.	DATE	4
created_by_user	ID of the Provider of Staff member who was logged in when the appointment was created. Blank if passwords not enabled.	CHARACTER	4
onpindboard	Flag to indicate if the appointment is on the Pinboard: - False= Appointment is not on the Pinboard - True= Appointment is on the Pinboard	BIT	1
origin_date	Date of appointment when the appointment was created.	DATE	4
origin_start_hour	Hour for time of appointment when the appointment was created.	TINYINT	1
origin_start_minute	Minutes for time of appointment when the appointment was created.	TINYINT	1
time_pattern	Bits to indicate the state of each time pattern checkbox for the length of the appointment (a checkbox for each time block). - Indicates chair time only if the checkbox is not marked - indicates only staff/assistant time if the checkbox is a slash (/) - indicates both provider and staff/assistant time if the checkbox is an X Added for G6 to allow for longer appointments.	CHARAC1TER	72

1.2 Modified Appointments

Stored Procedure: sp_appointments

Revision: 2

Inputs: BeginDate (DATE), EndDate (DATE), Only Changes Since Date/Time (TIMESTAMP)

Description: This stored procedure retrieves all modified appointments for a specified date range. When a Date/Time value is passed in for the 'Only Changes Since Date/Time' input, only those appointment records within the specified date range that have changed since the Date/Time that is passed in will be returned.

NOTE: A date range (BeginDate and EndDate) must be specified for this procedure. This allows the stored procedure to be used to either retrieve appointments that have a date within the specified date range or only "changed" appointments. In the case where you only wish to retrieve changed appointments (Deltas) and are not necessarily concerned with finding appointments within a certain date range, you should pass in a large date range such as a BeginDate of '01/01/1980' and EndDate of '01/01/2030' so that the *date range passed in is irrelevant and only the third parameter (the Date/Time passed in to get only the changed appointments) is used to filter the results. If you are not interested in retrieving changed appointments and only wish to have appointments returned that fall within a given date range, you must past in a Null value for the third parameter.*

Example: call admin.sp_appointments('08/01/2012','08/31/2012','08/15/2012 14:15:22')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
appointment_id	Appointment ID	INTEGER	4
appointment_date	Appointment date	DATE	4
operatory_id	Operatory ID in which the appointment is scheduled	CHARACTER	4
provider_id	Provider ID for this appointment	CHARACTER	4
provider_last_name	Provider last name	CHARACTER	21
provider_first_name	Provider first name	CHARACTER	16
provider_mi	Provider middle initial	CHARACTER	2
patient_id	Patient ID. This is 0 if the appointment is for a new patient.	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
patient_name	Patient name	CHARACTER	32
patient_phone	Patient's home phone number	CHARACTER	17
length	Appointment length	SMALLINT	2
reason	Appointment reason	CHARACTER	61
start_hour	Appointment start hour	TINYINT	1
start_minute	Appointment start minute	TINYINT	1
status_id	Appointment Status ID - 0=No Status - 1 through 10 = Status ID corresponding to the user-defined status specified in v_appt_status_codes - 100=Broken appointment - 101=Wait/Will Call (Unscheduled) appointment - -106=Completed - 150=Completed	TINYINT	1
status_description	Status description	CHARACTER	52
Amount	Appointment amount	MONEY	10
appt_broken_class	Flag for broken and unscheduled appointments: - 0= Scheduled appointment - 1=Unscheduled/broken appointment	TINYINT	1
broken_date	Used only for unscheduled/broken appointments. The date of the appointment	DATE	4

	when it is broken or moved to unscheduled; or if the appointment has never been scheduled, the current system date when appointment was unscheduled (wait/will call).		
appt_flag	Appointment "Schedule Type" flag: - 0= FIXED - 1=OPEN - 2=ASAP	TINYINT	1
auto_update_cc	Indicates whether or not Continuing Care (Recall) is automatically attached to the appointment according to the appointment reason: - 0= checkbox for "Use Reason to Auto Update CC" is not marked - 1="Use Reason to Auto Update CC" is marked	TINYINT	1
ref_by_type	Type of referral for new patient appointments. - 0= Referred by patient - 1= Referred by Dr/other	TINYINT	1
ref_id	Referral ID for new patient appointment. Links to a Referral Source record.	INTEGER	4
lab_case_appt	Flag to indicate if the appointment is related to a lab case: - 0=Appointment is not related to a lab case - 1=Appointment is related to a lab case.	BIT	1
create_date	Date the appointment was created.	DATE	4
created_by_user	ID of the Provider of Staff member who was logged in when the appointment was created. Blank if passwords not enabled.	CHARACTER	4
Pinboard	Flag to indicate if the appointment is on the Pinboard: - 0=Appointment is not on the Pinboard - 1=Appointment is on the Pinboard	TINYINT	1
Note	Appointment note	CHARACTER	5128
Servertime	Date and Time that this stored procedure was executed on the Dentrix server.	TIMESTAMP	8

1.3 All Patients' Appointment Information

Stored Procedure Name: sp_getallpatientsappointmentinfo
Revision: 1
Inputs: None
Description: This procedure returns the appointment information for all patients.

[CAUTION: This query could run slow depending on the number of patients]

Example: call admin.sp_getallpatientsappointmentinfo()

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHAR	26
first_name	Patient First Name	CHAR	16
next_appointment_date	Next Appointment Date	DATE	4
previous_appointment_date	Previous Appointment Date	DATE	4
number_of_missed_appointments	Number of Missed Appointments	INTEGER	4
last_missed_appointment_date	Last Missed Appointment Date	DATE	4

1.4 Patient's Appointment Information

Stored Procedure Name: sp_getpatientappointmentinfo

Revision: 1

Inputs: Patient GUID (CHARACTER)

Throws: PATIENT_NOT_FOUND

Description: This procedure returns the appointment information for a patient.

Example: call admin.sp_getpatientappointmentinfo('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	26
first_name	Patient First Name	CHARACTER	16
next_appointment_date	Next Appointment Date	DATE	4
previous_appointment_date	Previous Appointment Date	DATE	4
number_of_missed_appointments	Number of Missed Appointments	INTEGER	4
last_missed_appointment_date	Last Missed Appointment Date	DATE	4

1.5 Patient's Next Appointment Date and Time

Stored Procedure Name: sp_getpatientnextapptdatetime
Revision: 3
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the next appointment date and time for a patient. (This is mostly for use in other stored procedures)
Example: call admin.sp_getpatientnextapptdatetime('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
next_appointment_date	Next appointment Date	DATE	4
time_hour	Appointment begin Hour (1:00 PM is retrieved as 13)	TINYINT	1
time_minute	Appointment begin Minute	TINYINT	1

1.6 Patient's Previous Appointment Date and Time

Stored Procedure Name: sp_getpatientpreviousapptdatetime
Revision: 3
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the previous appointment date and time for a patient. (This is mostly used in other stored procedures).
Example: call admin.sp_getpatientpreviousapptdatetime('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
prev_date	Previous Appointment Date	DATE	4
time_hour	Appointment begin Hour (1:00 PM is retrieved as 13)	TINYINT	1
time_minute	Appointment begin Minute	TINYINT	1

1.7 Appointment Status Codes

Stored Procedure Name: v_appt_status_codes

Revision: 1

Description: This procedure returns the practice appointment status codes. A separate row will be returned for each appointment status code entered in the database.

Example: select* from admin.v_appt_status_codes

Column Name	Description	Type	Size
Defid	Status Definition ID	INTEGER	4
Descript	Status Description	CHARACTER	52

1.8 Appointment Types

View Name: v_appointment_types

Revision: 1

Description: This view returns a list of user-defined Appointment Types from Dentrix.

Example: select * from admin.v_appointment_types

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Appointment Type Definition ID	INTEGER	4
Descript	Appointment Type Description	CHARACTER	52

1.9 Appointment Check List

View Name: v_appointment_checklist

Revision: 1

Description: This view returns a list of user-defined Appointment Check List options from Dentrix.

Example: select * from admin.v_appointment_checklist

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Appointment Check List Item Definition ID	INTEGER	4
Descript	Appointment Check List Item Description	CHARACTER	52

1.10 Appointment Initial Reasons

View Name: v_initial_reasons

Revision: 1

Description: This view returns a list of user-defined Appointment Initial Reasons from Dentrix.
Example: select * from admin.v_initial_reasons

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Initial Appointment Reason Definition ID	INTEGER	4
Descript	Initial Appointment Reason Description	CHARACTER	52

1.11 Appointment Unit Size

Stored Procedure Name: sp_getappointmentunitsize
Revision: 2
Inputs: None
Description: This procedure returns the appointment book time block size setup for the practice.
Example: call admin.sp_getappointmentunitsize()

Column Name	Description	Type	Size
Timeblock	Appointment Book Time Block Size: - 1= 5 minutes - 2= 10 minutes - 3= 15 minutes - 4= 20 minutes - 6= 30 minutes	INTEGER	4

1.12 Appointment Amount Setting

Stored Procedure Name: sp_getapptamountsetting
Revision: 1
Inputs: None
Description: This procedure returns the value for the Appointment Amount setting in Dentrix.
Example: call admin.sp_getapptamountsetting()

Column Name	Description	Type	Size
settingvalue	Appointment Amount Setting Value: - 1= Always Calculate - 0= Allow Amount to be Entered/Fixed	INTEGER	4

1.13 Appointment Book Events

View Name: v_appt_book_events
Revision: 2

Description: This view retrieves Appointment Book events (non-appointment events scheduled in the Appointment Book). ***TIME CONVERSION NOTE: The returned start_time and end_time integer values can be converted to Time values as shown in the table below:***

Time Conversion Method	Time Conversion Example (result is 8:10 am)
< time value> = returned start_time / end_time value hour = *<time value>*/12 = hour (rounded) min = (*<time value>*%12) * 5 = 10	*98* = returned start_time / end_time value hour = 98/12 = 8 min = (98%12) * 5 = 10

Example: select * from admin.v_appt_book_events

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
event_id	Event ID	INTEGER	4
operatory_id	Operatory ID in which the Event is scheduled	CHARACTER	4
event_date	Event Date	DATE	4
start_time	Event start time (see 'Time Conversion Note' for this Table View above)	INTEGER	4
end_time	Event end time (see 'Time Conversion Note' for this Table View above)	INTEGER	4
close_op_flag	Flag to close the operatory for the entire day: - 0= Do not close operatory - 1= Close operatory	TINYINT	1
center_descript_flag	Flag to indicate that the description text should be centered for the event when the event is displayed on the Appointment Book: - 0= Do not center description - 1= Center description	TINYINT	1
color	Background color (HEX value) for the event when it is displayed on the appointment book.	BINARY	3
descript	Description for the Event.	CHARACTER	61
pinboard	Flag to indicate if the Event is on the Pinboard: - 0= Event is not on the Pinboard - 1= Event is on the Pinboard	TINYINT	1
orig_event_date	Date the Event was originally scheduled if the Event has been moved to the Pinboard. May be blank.	DATE	4
event_series_id	ID to link recurring Appointment Events records. May be 0.	INTEGER	4
note	Event note	CHARACTER	5128

1.14 Appointment Book Day Note

Stored Procedure Name: sp_apptbookdaynote

Revision: 3

Inputs: Appointment Book Date (DATE)

Throws: APPT_DAY_NOTE_NOT_FOUND

Description: This procedure returns the Appointment Book Day Note for the date (Appointment Book Date) that is passed in.

Example: call admin.sp_apptbookdaynote('11/18/2011')

Column Name	Description	Type	Size
appt_book_date	Appointment Book Date	DATE	4
appt_book_day_note	Appointment Book Day Note	CHARACTER	4000

1.15 Operatories

View Name: v_operatory

Revision: 1

Description: This view retrieves operatory information for the practice.

Example: select * from admin.v_operatory

Column Name	Description	Type	Size
op_id	Operatory ID	CHARACTER	4
op_title	Operatory Title	CHARACTER	21

1.16 Practice Default Schedule

Stored Procedure Name: sp_getpracticeschedule

Revision: 1

Inputs: None

Description: This procedure returns the default schedule for the practice.

Example: call admin.sp_getpracticeschedule()

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
practice_name	Practice Name	CHARACTER	60
sunday_start_time1	Sunday Start Time 1 (24 hour format)	TIME	4
sunday_end_time1	Sunday End Time 1 (24 hour format)	TIME	4
sunday_start_time2	Sunday Start Time 2 (24 hour format)	TIME	4
sunday_end_time2	Sunday End Time 2 (24 hour format)	TIME	4
sunday_start_time3	Sunday Start Time 3 (24 hour format)	TIME	4

sunday_end_time3	Sunday Start Time 3 (24 hour format)	TIME	4
monday_start_time1	Monday Start Time 1 (24 hour format)	TIME	4
monday_end_time1	Monday End Time 1 (24 hour format)	TIME	4
monday_start_time2	Monday Start Time 2 (24 hour format)	TIME	4
monday_end_time2	Monday End Time 2 (24 hour format)	TIME	4
monday_start_time3	Monday Start Time 3 (24 hour format)	TIME	4
monday_end_time3	Monday Start Time 3 (24 hour format)	TIME	4
tuesday_start_time1	Tuesday Start Time 1 (24 hour format)	TIME	4
tuesday_end_time1	Tuesday End Time 1 (24 hour format)	TIME	4
tuesday_start_time2	Tuesday Start Time 2 (24 hour format)	TIME	4
tuesday_end_time2	Tuesday End Time 2 (24 hour format)	TIME	4
tuesday_start_time3	Tuesday Start Time 3 (24 hour format)	TIME	4
tuesday_end_time3	Tuesday Start Time 3 (24 hour format)	TIME	4
wednesday_start_time1	Wednesday Start Time 1 (24 hour format)	TIME	4
wednesday_end_time1	Wednesday End Time 1 (24 hour format)	TIME	4
wednesday_start_time2	Wednesday Start Time 2 (24 hour format)	TIME	4
wednesday_end_time2	Wednesday End Time 2 (24 hour format)	TIME	4
wednesday_start_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
wednesday_end_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
thursday_start_time1	Thursday Start Time 1 (24 hour format)	TIME	4
thursday_end_time1	Thursday End Time 1 (24 hour format)	TIME	4
thursday_start_time2	Thursday Start Time 2 (24 hour format)	TIME	4
thursday_end_time2	Thursday End Time 2 (24 hour format)	TIME	4
thursday_start_time3	Thursday Start Time 3 (24 hour format)	TIME	4
thursday_end_time3	Thursday Start Time 3 (24 hour format)	TIME	4
friday_start_time1	Friday Start Time 1 (24 hour format)	TIME	4
friday_end_time1	Friday End Time 1 (24 hour format)	TIME	4
friday_start_time2	Friday Start Time 2 (24 hour format)	TIME	4
friday_end_time2	Friday End Time 2 (24 hour format)	TIME	4
friday_start_time3	Friday Start Time 3 (24 hour format)	TIME	4
friday_end_time3	Friday Start Time 3 (24 hour format)	TIME	4
saturday_start_time1	Saturday Start Time 1 (24 hour format)	TIME	4
saturday_end_time1	Saturday End Time 1 (24 hour format)	TIME	4
saturday_start_time2	Saturday Start Time 2 (24 hour format)	TIME	4
saturday_end_time2	Saturday End Time 2 (24 hour format)	TIME	4
saturday_start_time3	Saturday Start Time 3 (24 hour format)	TIME	4
saturday_end_time3	Saturday Start Time 3 (24 hour format)	TIME	4

1.17 Provider Default Schedule

Stored Procedure Name: sp_getproviderschedule
Revision: 1

Inputs: Provider ID (CHARACTER)
Throws: PROVIDER_NOT_FOUND
Description: This procedure returns the default schedule for a provider.
Example: call admin.sp_getproviderschedule('DDS1')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
prov_id	Provider ID	INTEGER	4
last_name	Provider Last Name	CHARACTER	21
first_name	Provider First Name	CHARACTER	16
sunday_start_time1	Sunday Start Time 1 (24 hour format)	TIME	4
sunday_end_time1	Sunday End Time 1 (24 hour format)	TIME	4
sunday_start_time2	Sunday Start Time 2 (24 hour format)	TIME	4
sunday_end_time2	Sunday End Time 2 (24 hour format)	TIME	4
sunday_start_time3	Sunday Start Time 3 (24 hour format)	TIME	4
sunday_end_time3	Sunday Start Time 3 (24 hour format)	TIME	4
monday_start_time1	Monday Start Time 1 (24 hour format)	TIME	4
monday_end_time1	Monday End Time 1 (24 hour format)	TIME	4
monday_start_time2	Monday Start Time 2 (24 hour format)	TIME	4
monday_end_time2	Monday End Time 2 (24 hour format)	TIME	4
monday_start_time3	Monday Start Time 3 (24 hour format)	TIME	4
monday_end_time3	Monday Start Time 3 (24 hour format)	TIME	4
tuesday_start_time1	Tuesday Start Time 1 (24 hour format)	TIME	4
tuesday_end_time1	Tuesday End Time 1 (24 hour format)	TIME	4
tuesday_start_time2	Tuesday Start Time 2 (24 hour format)	TIME	4
tuesday_end_time2	Tuesday End Time 2 (24 hour format)	TIME	4
tuesday_start_time3	Tuesday Start Time 3 (24 hour format)	TIME	4
tuesday_end_time3	Tuesday Start Time 3 (24 hour format)	TIME	4
wednesday_start_time1	Wednesday Start Time 1 (24 hour format)	TIME	4
wednesday_end_time1	Wednesday End Time 1 (24 hour format)	TIME	4
wednesday_start_time2	Wednesday Start Time 2 (24 hour format)	TIME	4
wednesday_end_time2	Wednesday End Time 2 (24 hour format)	TIME	4
wednesday_start_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
wednesday_end_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
thursday_start_time1	Thursday Start Time 1 (24 hour format)	TIME	4
thursday_end_time1	Thursday End Time 1 (24 hour format)	TIME	4
thursday_start_time2	Thursday Start Time 2 (24 hour format)	TIME	4
thursday_end_time2	Thursday End Time 2 (24 hour format)	TIME	4
thursday_start_time3	Thursday Start Time 3 (24 hour format)	TIME	4
thursday_end_time3	Thursday Start Time 3 (24 hour format)	TIME	4
friday_start_time1	Friday Start Time 1 (24 hour format)	TIME	4
friday_end_time1	Friday End Time 1 (24 hour format)	TIME	4

friday_start_time2	Friday Start Time 2 (24 hour format)	TIME	4
friday_end_time2	Friday End Time 2 (24 hour format)	TIME	4
friday_start_time3	Friday Start Time 3 (24 hour format)	TIME	4
friday_end_time3	Friday Start Time 3 (24 hour format)	TIME	4
saturday_start_time1	Saturday Start Time 1 (24 hour format)	TIME	4
saturday_end_time1	Saturday End Time 1 (24 hour format)	TIME	4
saturday_start_time2	Saturday Start Time 2 (24 hour format)	TIME	4
saturday_end_time2	Saturday End Time 2 (24 hour format)	TIME	4
saturday_start_time3	Saturday Start Time 3 (24 hour format)	TIME	4
saturday_end_time3	Saturday Start Time 3 (24 hour format)	TIME	4

1.18 Operatory Default Schedule

Stored Procedure Name: sp_getoperatoryschedule

Revision: 1

Inputs: Operatory ID (CHARACTER)

Throws: OPERATORY_NOT_FOUND

Description: This procedure returns the default schedule for an operatory.

Example: call admin.sp_getoperatoryschedule('OP-1')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
op_id	Operatory ID	INTEGER	4
op_title	Operatory Title	CHARACTER	21
sunday_start_time1	Sunday Start Time 1 (24 hour format)	TIME	4
sunday_end_time1	Sunday End Time 1 (24 hour format)	TIME	4
sunday_start_time2	Sunday Start Time 2 (24 hour format)	TIME	4
sunday_end_time2	Sunday End Time 2 (24 hour format)	TIME	4
sunday_start_time3	Sunday Start Time 3 (24 hour format)	TIME	4
sunday_end_time3	Sunday Start Time 3 (24 hour format)	TIME	4
monday_start_time1	Monday Start Time 1 (24 hour format)	TIME	4
monday_end_time1	Monday End Time 1 (24 hour format)	TIME	4
monday_start_time2	Monday Start Time 2 (24 hour format)	TIME	4
monday_end_time2	Monday End Time 2 (24 hour format)	TIME	4
monday_start_time3	Monday Start Time 3 (24 hour format)	TIME	4
monday_end_time3	Monday Start Time 3 (24 hour format)	TIME	4
tuesday_start_time1	Tuesday Start Time 1 (24 hour format)	TIME	4
tuesday_end_time1	Tuesday End Time 1 (24 hour format)	TIME	4
tuesday_start_time2	Tuesday Start Time 2 (24 hour format)	TIME	4
tuesday_end_time2	Tuesday End Time 2 (24 hour format)	TIME	4
tuesday_start_time3	Tuesday Start Time 3 (24 hour format)	TIME	4
tuesday_end_time3	Tuesday Start Time 3 (24 hour format)	TIME	4

wednesday_start_time1	Wednesday Start Time 1 (24 hour format)	TIME	4
wednesday_end_time1	Wednesday End Time 1 (24 hour format)	TIME	4
wednesday_start_time2	Wednesday Start Time 2 (24 hour format)	TIME	4
wednesday_end_time2	Wednesday End Time 2 (24 hour format)	TIME	4
wednesday_start_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
wednesday_end_time3	Wednesday Start Time 3 (24 hour format)	TIME	4
thursday_start_time1	Thursday Start Time 1 (24 hour format)	TIME	4
thursday_end_time1	Thursday End Time 1 (24 hour format)	TIME	4
thursday_start_time2	Thursday Start Time 2 (24 hour format)	TIME	4
thursday_end_time2	Thursday End Time 2 (24 hour format)	TIME	4
thursday_start_time3	Thursday Start Time 3 (24 hour format)	TIME	4
thursday_end_time3	Thursday Start Time 3 (24 hour format)	TIME	4
friday_start_time1	Friday Start Time 1 (24 hour format)	TIME	4
friday_end_time1	Friday End Time 1 (24 hour format)	TIME	4
friday_start_time2	Friday Start Time 2 (24 hour format)	TIME	4
friday_end_time2	Friday End Time 2 (24 hour format)	TIME	4
friday_start_time3	Friday Start Time 3 (24 hour format)	TIME	4
friday_end_time3	Friday Start Time 3 (24 hour format)	TIME	4
saturday_start_time1	Saturday Start Time 1 (24 hour format)	TIME	4
saturday_end_time1	Saturday End Time 1 (24 hour format)	TIME	4
saturday_start_time2	Saturday Start Time 2 (24 hour format)	TIME	4
saturday_end_time2	Saturday End Time 2 (24 hour format)	TIME	4
saturday_start_time3	Saturday Start Time 3 (24 hour format)	TIME	4
saturday_end_time3	Saturday Start Time 3 (24 hour format)	TIME	4

1.19 Practice Schedule Exceptions

Stored Procedure Name: sp_getpracticeschedexceptions

Revision: 1

Inputs: BeginDate (DATE), EndDate (DATE)

Description: This procedure returns schedule exceptions for the practice for the date range specified.

Example: call admin.sp_getpracticeschedexceptions('05/01/2012','05/31/2012')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
practice_name	Practice Name	CHARACTER	60
sched_exception_date	Practice Schedule Exception Date	DATE	4
closed_flag	Closed / Holiday Flag. Indicates whether the Practice is flagged as being closed for Holiday or other reasons. - 1=Not Closed - 2=Closed	TINYINT	1

	- 3=Holiday - 4=Closed and Holiday		
start_time1	Start Time 1 (24 hour format)	TIME	4
end_time1	End Time 1 (24 hour format)	TIME	4
start_time2	Start Time 2 (24 hour format)	TIME	4
end_time2	End Time 2 (24 hour format)	TIME	4
start_time3	Start Time 3 (24 hour format)	TIME	4
end_time3	Start Time 3 (24 hour format)	TIME	4

1.20 Provider Schedule Exceptions

Stored Procedure Name: sp_getprovschedexceptions

Revision: 1

Inputs: Provider ID (CHARACTER), BeginDate (DATE), EndDate (DATE)

Throws: PROVIDER_NOT_FOUND

Description: This procedure returns schedule exceptions for a provider for the date range specified.

Example: call admin.sp_getprovschedexceptions('DDS1','05/01/2012','05/31/2012')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
prov_id	Provider ID	INTEGER	4
last_name	Provider Last Name	CHARACTER	21
first_name	Provider First Name	CHARACTER	16
sched_exception_date	Provider Schedule Exception Date	DATE	4
vacation_flag	Vacation Flag. Indicates whether the Provider is flagged as being out/on vacation for the day. - 0=Not on Vacation - 1=On Vacation	TINYINT	1
start_time1	Start Time 1 (24 hour format)	TIME	4
end_time1	End Time 1 (24 hour format)	TIME	4
start_time2	Start Time 2 (24 hour format)	TIME	4
end_time2	End Time 2 (24 hour format)	TIME	4
start_time3	Start Time 3 (24 hour format)	TIME	4
end_time3	Start Time 3 (24 hour format)	TIME	4

1.21 Operatory Schedule Exceptions

Stored Procedure Name: sp_getoperatoryschedexceptions

Revision: 1

Inputs: Operatory ID (CHARACTER), BeginDate (DATE), EndDate (DATE)

Throws: OPERATORY_NOT_FOUND

Description: This procedure returns schedule exceptions for an operatory for the date range specified.

Example: call admin.sp_getoperatoryschedexceptions('OP-1','05/01/2012','05/31/2012')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
op_id	Operatory ID	INTEGER	4
op_title	Operatory Title	CHARACTER	21
sched_exception_date	Provider Schedule Exception Date	DATE	4
start_time1	Start Time 1 (24 hour format)	TIME	4
end_time1	End Time 1 (24 hour format)	TIME	4
start_time2	Start Time 2 (24 hour format)	TIME	4
end_time2	End Time 2 (24 hour format)	TIME	4
start_time3	Start Time 3 (24 hour format)	TIME	4
end_time3	Start Time 3 (24 hour format)	TIME	4

1.22 Provider Time Blocks (Default)

Stored Procedure Name: sp_getprovtimeblockdefault

Revision: 1

Inputs: Provider ID (CHARACTER)

Throws: PROVIDER_NOT_FOUND

Description: This procedure returns the default time blocks setup for a specified provider.

Example: call admin.sp_getprovtimeblockdefault('DDS1')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
prov_id	Provider ID	INTEGER	4
time_block_name	Time Block Name	CHARACTER	13
time_block_color	Time Block Color (HEX value)	BINARY	3
assign_op	Assigned Operatory	CHARACTER	4
block_appt_type	Block Appt Type - 0=All - 1=General - 2=High Production - 3=Medium Production - 4=Low Production	TINYINT	1
sunday_start_time	Sunday Start Time (24 hour format)	TIME	4
sunday_end_time	Sunday End Time (24 hour format)	TIME	4
monday_start_time	Monday Start Time (24 hour format)	TIME	4
monday_end_time	Monday End Time (24 hour format)	TIME	4
tuesday_start_time	Tuesday Start Time (24 hour format)	TIME	4
tuesday_end_time	Tuesday End Time (24 hour format)	TIME	4

wednesday_start_time	Wednesday Start Time (24 hour format)	TIME	4
wednesday_end_time	Wednesday End Time (24 hour format)	TIME	4
thursday_start_time	Thursday Start Time (24 hour format)	TIME	4
thursday_end_time	Thursday End Time (24 hour format)	TIME	4
friday_start_time	Friday Start Time (24 hour format)	TIME	4
friday_end_time	Friday End Time (24 hour format)	TIME	4
saturday_start_time	Saturday Start Time (24 hour format)	TIME	4
saturday_end_time	Saturday End Time (24 hour format)	TIME	4

1.23 Provider Time Blocks (Specified Week)

Stored Procedure Name: sp_getprovtimeblockschedule

Revision: 1

Inputs: Provider ID (CHARACTER)

Throws: PROVIDER_NOT_FOUND

Description: This procedure returns the time blocks setup for a given provider for a specified week relative to the date that is passed in.

Example: call admin.sp_getprovtimeblockschedule('DDS1','05/05/2015')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
prov_id	Provider ID	INTEGER	4
time_block_name	Time Block Name	CHARACTER	13
time_block_color	Time Block Color (HEX value)	BINARY	3
assign_op	Assigned Operatory	CHARACTER	4
block_appt_type	Block Appt Type - 0=All - 1=General - 2=High Production - 3=Medium Production - 4=Low Production	TINYINT	1
week_start_date	Week Start Date for Returned Time Blocks	DATE	4
sunday_start_time	Sunday Start Time (24 hour format)	TIME	4
sunday_end_time	Sunday End Time (24 hour format)	TIME	4
monday_start_time	Monday Start Time (24 hour format)	TIME	4
monday_end_time	Monday End Time (24 hour format)	TIME	4
tuesday_start_time	Tuesday Start Time (24 hour format)	TIME	4
tuesday_end_time	Tuesday End Time (24 hour format)	TIME	4
wednesday_start_time	Wednesday Start Time (24 hour format)	TIME	4
wednesday_end_time	Wednesday End Time (24 hour format)	TIME	4
thursday_start_time	Thursday Start Time (24 hour format)	TIME	4
thursday_end_time	Thursday End Time (24 hour format)	TIME	4
friday_start_time	Friday Start Time (24 hour format)	TIME	4

friday_end_time	Friday End Time (24 hour format)	TIME	4
saturday_start_time	Saturday Start Time (24 hour format)	TIME	4
saturday_end_time	Saturday End Time (24 hour format)	TIME	4

2. Balance and Payment Agreement

2.1 Patient's Balance

Stored Procedure Name: sp_getpatientbalance
Revision: 2
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the patient balances (30, 60, 90, 91 plus) and the last payment details.
Example: call admin.sp_getpatientbalance('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
balance_0_30_days	Patient balance,0 -30 days	MONEY	10
balance_31_60_days	Patient balance,31 -60 days	MONEY	10
balance_61_90_days	Patient balance,61-90 days	MONEY	10
balance_91_ plus_days	Patient balance,91 plus days	MONEY	10
last_payment_date	Patient Last Payment Date	DATE	4
last_ payment _amount	Patient Last Payment Amount	MONEY	32
last_ payment _type	Patient Last Payment Type	TINYINT	1
last_ payment _desc	Payment Type Description	CHARACTER	52

2.2 Guarantor's Balance

View Name: v_guarantor_balance
Version: 1
Description: This view retrieves all the guarantor records and their balances.
Example: select * from admin.v_guarantor_balance

Column Name	Description	Type	Size
guar_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
last_name	Last name	CHARACTER	21

first_name	First name	CHARACTER	16
mi	Middle initial	CHARACTER	2
balance_0_30_days	Balance, 0-30 days	MONEY	10
balance_31_60_days	Balance, 31-60 days	MONEY	10
balance_61_90_days	Balance, 61-90 days	MONEY	10
balance_91_plus_days	Balance, >90 days	MONEY	10
last_payment_received_date	Last payment received date	DATE	4
last_payment_amount	Last payment amount	MONEY	10
email	Email address	CHARACTER	60
mobile_phone	Pager	CHARACTER	17
birth_date	Birth date	DATE	4
status	Status	TINYINT	1
social_sec_num	Social Security Number	CHARACTER	10
billing_type_description	Billing type description (such as Standard Billing – Finance Charges)	CHARACTER	52

2.3 Guarantor's Payment Agreement

Stored Procedure Name: sp_getguarpmtagreement

Revision: 1

Inputs: Guarantor GUID (CHARACTER)

Description: This procedure returns the payment agreement information for a guarantor / family account.

Example: call admin.sp_getguarpmtagreement('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
guarantor_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
last_name	Guarantor Last Name	CHARACTER	21
first_name	Guarantor First Name	CHARACTER	16
last_pmt_date	Last Payment Date	DATE	4
last_pmt_amt	Last Payment Amount	MONEY	10

last_pmt_desc	Last Payment Description	CHARACTER	52
next_pmt_date	Next Payment Date	DATE	4
next_pmt_amt	Next Payment Amount	MONEY	10
rem_balance	Payment Agreement Remaining Balance	MONEY	10
past_due_amt	Amount Past Due	MONEY	10

3. Dental Labs

3.1 Dental Lab Information

View Name: v_lab
Revision: 1
Description: This view retrieves dental labs setup in Dentrix.
Example: select * from admin.v_lab

Column Name	Description	Type	Size
lab_id	Lab ID	TIMESTAMP	8
Name	Lab Name	CHARACTER	32
address_line1	Address Line 1	CHARACTER	32
address_line2	Address Line 2	CHARACTER	32
City	City	CHARACTER	26
State	State	CHARACTER	21
zip_code	Zip Code	CHARACTER	15
phone	Phone	CHARACTER	17
phone_ext	Phone Extension	CHARACTER	5
contact	Lab Contact	CHARACTER	32
email	Email	CHARACTER	60
Fax	Fax	CHARACTER	17

3.2 Dental Lab Cases

View Name: v_lab_case
Revision: 1
Description: This view retrieves patient's dental lab cases.
Example: select * from admin.v_lab_case

Column Name	Description	Type	Size
automodifiedtimestamp	Modified Date/Time Stamp	TIMESTAMP	8
caseid	Lab Case ID	CHARACTER	32

labid	Lab ID	CHARACTER	32
patid	Patient ID	CHARACTER	32
createdate	Lab Case Created Date	CHARACTER	26
finisheddate	Lab Case Finished Date	CHARACTER	21
categoryid	Category ID (Links to the Lab Case Manager Definitions table)	CHARACTER	15
provid	Provider ID	CHARACTER	5
casenum	Lab Case Assigned Case #	CHARACTER	32
caseguid	Lab Case GUID	CHARACTER	60
flags	Flags Lab Case as archived. - 0=Not Archived - 1=Archived	CHARACTER	17

4. Insurance and Employers

4.1 Patient's Dental Insurance Information

View Name: v_patient_insurance
Revision: 1
Description: This view returns patients' primary and secondary dental insurance information.
Example: select * from admin.v_patient_insurance

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
primary_insured_id	Links to the insured_id (in 'v_insured' table view) for patient's primary dental insurance.	INTEGER	4
primary_insured_last_name	Primary Subscriber Last Name	CHARACTER	21
primary_insured_first_name	Primary Subscriber First Name	CHARACTER	16
secondary_insured_id	Links to the insured_id (in v_insured table view) for patient's secondary dental insurance.	INTEGER	4
secondary_insured_last_name	Secondary subscriber Last Name	CHARACTER	21
secondary_insured_first_name	Secondary subscriber First Name	CHARACTER	16
primary_insurance_carrier_id	Primary Insurance Carrier ID	INTEGER	4
primary_insurance_carrier_name	Primary Insurance Carrier Name	CHARACTER	32
secondary_insurance_carrier_id	Primary Insurance Carrier ID	INTEGER	4
seconday_insurance_carrier_name	Secondary Insurance Carrier Name	CHARACTER	32

4.2 All Patients' Dental Insurance Information

Stored Procedure Name: sp_getallpatientsinsuranceinfo
Revision: 2

Inputs: None

Description: This procedure returns the primary and secondary dental insurance information for all patients.
[CAUTION: This query could run slowly depending on the number of patients.]

Example: call admin.sp_getallpatientsinsuranceinfo()

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
primary_insured_id	Links to the insured_rec for patient's primary dental insurance.	INTEGER	4
primary_insured_last_name	Primary Subscriber Last Name	CHARACTER	21
primary_insured_first_name	Primary Subscriber First Name	CHARACTER	16
secondary_insured_id	Links to the insured_rec for patient's primary dental insurance.	INTEGER	4
secondary_insured_last_name	Secondary subscriber Last Name	CHARACTER	21
secondary_insured_first_name	Secondary subscriber First Name	CHARACTER	16
primary_insurance_carrier_id	Primary Insurance Carrier ID	INTEGER	4
primary_insurance_carrier_name	Primary Insurance Carrier Name	CHARACTER	32
secondary_insurance_carrier_id	Primary Insurance Carrier ID	INTEGER	4
seconday_insurance_carrier_name	Secondary Insurance Carrier Name	CHARACTER	32

4.3 Insurance Plans

View Name: v_insurance_plans

Revision: 1

Description: This view retrieves insurance plans setup in Dentrix.

Example: select * from admin.v_insurance_plans

Column Name	Description	Type	Size
ins_id	Insurance Carrier ID	INTEGER	4
ins_co_name	Insurance Carrier Name	CHARACTER	32

group_name	Insurance Plan Group Name	CHARACTER	32
group_number	Insurance Plan Group Number	CHARACTER	25
union_number	Number for Union Members (ie. Local 99).	CHARACTER	7
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
city	City	CHARACTER	26
state	State	CHARACTER	21
zipcode	ZIP Code	CHARACTER	15
Phone	Insurance Plan Phone	CHARACTER	17
phone_ext	Insurance Plan Phone extension	CHARACTER	5
contact	Contact Name	CHARACTER	31
renew_month	Benefit renewal month	TINYINT	1
last_update	Last Update	DATE	4
max_cov_person	Annual maximum benefit, individual (used for Dental insurance only).	MONEY	10
max_cov_fam	Annual maximum benefit, family (used for Dental insurance only).	MONEY	10
time_limit	Claim deadline in number of days. The user can select 1 through 6, days, weeks, months or years. Used to calculate an "expires" date on the Insurance Claims and reports (Not used for medical insurance plans).	SMALLINT	2
source_of_payment	Source of Payment: - 0=None - 67=Medicare (not used for Dental Ins.) - 68=Medicaid - 70=Commercial Ins. Co. - 71=Blue Cross/Blue Shield - 72=TRICARE/CHAMPUS	TINYINT	1
ins_flag	Insurance Type Flag - 0=Dental - 1=Medical	TINYINT	1

std_ded_perperson	Annual individual standard deductible (Not used for medical insurance plans).	MONEY	10
prv_ded_perperson	Annual individual preventive deductible (Not used for medical insurance plans).	MONEY	10
other_ded_perperson	Annual individual other deductible (Not used for medical insurance plans)	MONEY	10
std_ded_perperson_lt	Lifetime individual standard deductible (Not used for medical insurance plans).	MONEY	10
prv_ded_perperson_lt	Lifetime individual preventive deductible (Not used for medical insurance plans).	MONEY	10
other_ded_perperson_lt	Lifetime individual other deductible (Not used for medical insurance plans).	MONEY	10
std_ded_perfam	Annual family standard deductible (Not used for medical insurance plans).	MONEY	10
prv_ded_perfam	Annual family preventive deductible (Not used for medical insurance plans).	MONEY	10
other_ded_perfam	Annual family other deductible (Not used for medical insurance plans).	MONEY	10
do_not_bill_ins_flag	Flag for “Do Not Bill to Dental Insurance” (Not used for medical insurance plans). - True=Do Not Bill to Dental Insurance - False=Allow Dental Insurance to be Billed	BIT	1
diag_print_flag	Flag for “Do Not Include Dental Diagnostic Codes” (Not used for medical insurance plans). - 1=Do Not Include Dental Diagnostic Codes - 0=All Dental Diagnostic Codes to be Included	BIT	1
do_not_include_group	Flag for “Do Not Include Group Plan Name”.	TINYINT	1
national_plan_id	National plan ID (NPI #).	CHARACTER	21

4.4 Insurance Plan Notes

View Name: v_insurance_plan_notes

Revision: 1
Description: This view returns insurance plan notes
Example: select * from admin.v_insurance_plan_notes

Column Name	Description	Type	Size
carrier_id	Insured Carrier ID	INTEGER	4
note_text	Insurance Carrier Note	CHARACTER	3,999

4.5 Insured Subscribers

View Name: v_insured
Revision: 1
Description: This view returns insured subscriber information.
Example: select * from admin.v_insured

Column Name	Description	Type	Size
insured_id	Insured Subscriber's Insured ID for Primary / Secondary Dental or Medical Insurance	INTEGER	4
ins_plan_id	Insurance Carrier ID	CHARACTER	32
ins_party_id	Insured Subscriber's Patient ID	INTEGER	4
family_benefits	Current year's family benefits applied amount. Not used for medical insurance.	MONEY	4
id_num	Subscriber ID #	CHARACTER	25
ins_type	Insurance Type flag: - 0=Dental - 2=Medical	TINYINT	1
family_std_ded_met	Current year's annual family standard deductible met. Not used for medical insurance plans.	MONEY	4
family_prv_ded_met	Current year's annual family preventive deductible met. Not used for medical insurance plans.	MONEY	4
family_oth_ded_met	Current year's annual family other deductible met. Not used for medical insurance plans.	MONEY	4

4.6 Employer

View Name: v_employers

Revision: 1

Description: This view returns Employer information.

Example: select * from admin.v_employers

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
employer_id	Employer ID	INTEGER	4
employer_name	Employer Name	CHARACTER	32
address_line1	Address Line 1	CHARACTER	31
address_line2	Address Line 2	CHARACTER	31
City	City	CHARACTER	26
State	State	CHARACTER	16
zip_code	Zip Code	CHARACTER	15
Phone	Phone	CHARACTER	17
address_line1	Address Line 1	CHARACTER	31

5. Ledger and Treatment Plan

5.1 Ledger Transactions

View Name: v_proclog

Revision: 0

Description: This view returns all posted transactions for a patient stored in the Procedure Log table (treatment planned procedures, completed procedures, adjustments and payments).

Example: select * from admin.v_proclog

Column Name	Description	Type	Size
automodifiedtimestamp	Modified Date/Time Stamp	TIMESTAMP	8
patid	Patient ID	INTEGER	4
procid	Transaction ID	INTEGER	4
guarid	Guarantor ID	INTEGER	4
chartstatus	Flag to indicate type of transaction: - 90 = payment, insurance payment, adjustment, special adjustment, finance charge, late charge, balance forward, initial balance (a transaction only used for the Ledger, not used for the Chart) - 100 = existing work by another provider—a transaction only used for the Chart, not used for the Ledger - 101 = existing work by the current provider—a transaction only used for the Chart, not used for the Ledger - 102 = completed procedure—a transaction used for both the Chart and the Ledger - 105 = treatment planned procedure—a transaction only used for the Chart, not used for the Ledger (except when listing treatment planned procedures from Options | Treatment Plan) - 109 = condition or diagnosis—a transaction only used for the Chart, not used for the Ledger	TINYINT	1

	- 110 = primary/permanent change—a transaction only used for the Chart, not used for the Ledger. Not used as of G5. - 112 = treatment planned procedure that is rejected or an alternate treatment plan that is not recommended—a transaction only used for the Treatment Planner, not used for the Chart or Ledger		
procdate	Transaction / Procedure Date. Defaults to be the same as the current system date at the time that the transaction is posted, but may be changed by the user.	DATE	4
proccodeid	ID for Procedure Code or Patient: - If ChartStatus is 110, this will be 0. - If ChartStatus is not 90 or 110, this links to proccodeid in *v_proccodes* table view. - If ChartStatus is 90, this may be 0 or it links to patien_id in v_patient table view. - When this is not 0 (it is an account payment or an adjustment attached to a patient), the patid and guarId for this transaction record will both be for the guarantor's Patients record.	INTEGER	4
preauthid	Insurance Claim Pre-Authorization ID	INTEGER	4
claimid	Insurance Claim ID	INTEGER	4
proclogclass	Ledger class for the following (where ChartStatus is 90): - 0 = balance forward or an initial balance - 1 = account payment - 2 = adjustment - 3 = insurance payment - 4 = finance charge - 7 = special adjustment - 8 = late charge	TINYINT	1
proclogorder	For payment type or adjustment type; or to flag balance forward or initial	TINYINT	1

	balance; or for treatment plan visit number. May be 0. - If ChartStatus = 90 (for Ledger) and ProcLogClass = 1. Links to Payment Types table view (see def_id column in v_payment_types) for account payment type. - If ChartStatus is 90 (for Ledger) and ProcLogClass is 2. Links to Adjustment Types table view (see def_id column in v_adjustment_types) for adjustment type. - If ChartStatus is 90 (for Ledger) and ProcLogClass is 0: ▪ 0 ProcLogOrder indicates a balance forward record ▪ 1 ProcLogOrder indicates an initial balance record - If ChartStatus is 102 (completed procedure), 105 (treatment planned procedure), or 112 (treatment planned procedure not recommended): A number in this field indicates the visit number for the procedure in the treatment plan case.		
provid	Provider ID. Links to Providers table view (see provider_id column in v_provider) for the transaction provider. For procedures, this is the treatment planning provider when a procedure is treatment planned; and when the procedure is completed, this is changed to store the rendering provider.	CHARACTER	4
history	Flag to indicate that transaction is in History - 1 = Transaction is in History - 0 = Transaction is not in History	BIT	1
amt	Transaction Amount. May be 0.00.	MONEY	10
amtpriminspaid	Primary Insurance Paid Amount	MONEY	10
amtsecinspaid	Secondary Insurance Paid Amount	MONEY	10

entrydate	Transaction Entry date. Defaults to be the same as the current system date at the time the transaction is posted; cannot be changed by the user.	DATE	4
toothrangestart	Tooth number or beginning of tooth number range for a procedure. May be blank. See 'Dentrix Tooth Numbering.pdf' in the 'Misc Docs' folder in *DentrixSDK_2.1.zip* for how tooth numbers are stored.	TINYINT	1
toothrangeend	Ending of tooth number range for a procedure. May be blank.	TINYINT	1
surfacestring	- If ChartStatus is not 90, the surface string as it should be displayed for a procedure (the individual surfaces combined) - If this is a payment, the bank number. May be blank.	BINARY	20
surfm	- If this is a procedure with "Surface" selected for treatment area for the procedure code, this flags if the "Mesial" surface is selected. - If this is a procedure with "Quadrant" selected for treatment area for the procedure code, this is for the selected quadrant: o 1= UR o 2= UL o 3= LL o 4= LR - If this is a medical insurance payment, flags if "Include Payment on Dental Claims" checkbox is marked.	TINYINT	1
surfo	If this is a procedure with "Surface" selected for treatment area for the procedure code, flags if the "Incisal/Occlusal" surface is selected (1) or not (0).	BIT	1
surfd	If this is a procedure with "Surface" selected for treatment area for the procedure code, flags if the "Distal" surface is selected.	BIT	1

surff	If this is a procedure with "Surface" selected for treatment area for the procedure code, this flags if the "Lingual" surface is selected (1) or not (0).	BIT	1
surfl	If this is a procedure with "Surface" selected for treatment area for the procedure code, flags if the "Facial/Buccal" surface is selected (1) or not (0).	BIT	1
surf5	If this is a procedure with "Surface" selected for treatment area for the procedure code, flags if the "Class 5" surface is selected (1) or not (0).	BIT	1
invalidasofflagstpdate	- If ChartStatus is 109 for a condition and the condition is invalidated: The date the condition was invalidated. - If ChartStatus is 90 and ProcLogClass is 1 for an account payment: Flags whether or not the payment applies to the account's payment agreement (1 indicates it does). G5 change. - If ChartStatus is 102 (completed procedure): Date the procedure was treatment planned, if it was first treatment planned. When a treatment planned procedure is completed, this becomes equal to ProcDate (procedure date), and then ProcDate is updated to be the current system date for the date the procedure was completed. May be 0.	DATE	4
medproctype	Flags if a procedure is medical cross coded.	BIT	11
donotbillinsflag	Flag for the "Do Not Bill to Dental Insurance" checkbox. - 0= checkbox is not marked - 1= checkbox is marked Defaults to equal DoNotBillInsFlag setting specified for the procedure code when the procedure is posted.	BIT	1

diag	Flags if dental diagnostic codes are attached to the procedure (1) or not (0).	BIT	1
refid	Links to Referral Sources (ref_id column in sp_getrefrralsource stored procedure) for the referral that this procedure is referred to or referred by. May be 0.	INTEGER	4
reftype	If RefId is not 0, flags a “referred by” (0) or a “referred to” (1).	TINYINT	1
txcaseid	Links to Treatment Plan Cases table if the procedure is treatment planned. May be 0.	INTEGER	4
startcompdatereq	Flag for start and completed dates required for procedure (defaults from procedure code). - 0= the checkbox is not marked - 1= the checkbox is marked	TINYINT	1
completiondate	Completed date for when a start and completed date is required for the procedure.	DATE	4
startdate	Start date for when a start and completed date is required for the procedure.	DATE	4

5.2 Account Payments

Stored Procedure Name: sp_getaccountpayments

Revision: 2

Inputs: Patient GUID (CHARACTER), BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the patient and insurance payments for all family members on an account (according to the Patient ID used for the input) for a specified date range. The byCreateDate is set to 1 to retrieve payments by entry date. This flag is set to 0 to retrieve payments by the procedure date.

Example: call admin.sp_getaccountpayments(‘045e8d6e-a74d-4bda-a9ec-0db9609ee61a’,’01/01/2000’,’01/01/2010’,0)

Column Name	Description	Type	Size

patient_id	Patient ID (will be 0 if the payment was applied to the Family rather than an individual patient).	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
guarantor_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
patient_last_name	Patient Last Name (will be "Family" if the payment was applied to the Family rather than an individual patient).	CHARACTER	21
patient_first_name	Patient First Name (will be "Family" if the payment was applied to the Family rather than an individual patient).	CHARACTER	16
guar_last_name	Guarantor Last Name	CHARACTER	21
guar_first_name	Guarantor First Name	CHARACTER	16
description	Payment Description/Type	CHARACTER	52
ins_plan_name	Insurance Plan Name (may be blank)	CHATACTER	52
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
proc_date	Procedure Date	DATE	4
Amt	Amount	MONEY	10
check_num	Check Number	STRING	21
branch_num	Branch Number	STRING	20

5.3 Patient's Adjustments

Stored Procedure Name: sp_getpatientadjustments

Revision: 2

Inputs: Patient GUID (CHARACTER), BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the adjustments for all family members on an account (according to the Patient ID used for the input) for a specified date range. The byCreateDate is set to 1 to retrieve adjustments by entry date. This flag is set to 0 to retrieve adjustments by the procedure date.

Example: call admin.sp_getpatientadjustments('045e8d6e-a74d-4bda-a9ec-0db9609ee61a','01/01/2000','01/01/2010',0)

Column Name	Description	Type	Size
patient_id	Patient ID (will be 0 if the adjustment was applied to the Family rather than an individual patient).	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
guarantor_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
last_name	Patient Last Name (will be “Family” if the adjustment was applied to the Family rather than an individual patient).	CHARACTER	21
first_name	Patient First Name (will be “Family” if the adjustment was applied to the Family rather than an individual patient).	CHARACTER	16
guar_last_name	Guarantor Last Name	CHARACTER	21
guar_first_name	Guarantor First Name	CHARACTER	16
description	Adjustment Description/Type	CHARACTER	52
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
proc_date	Procedure Date	DATE	4
amt	Amount	MONEY	10

5.4 Patient’s Completed Procedures

Stored Procedure Name: sp_getpatientprocedures

Revision: 3

Inputs: Patient GUID (CHARACTER), BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the completed procedures for a patient for a specified date range. The byCreateDate is set to 1 to retrieve the procedures by entry date. This flag is set to 0 to retrieve the procedures by the procedure date.

Example: call admin.sp_getpatientprocedures(‘045e8d6e-a74d-4bda-a9ec-0db9609ee61a’,’01/01/2000’,’01/01/2010’,0)

Column Name	Description	Type	Size

patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
guar_last_name	Guarantor Last Name	CHARACTER	21
guar_first_name	Guarantor First Name	CHARACTER	16
proc_code	ADA Procedure Code	CHARACTER	10
description	Procedure Description	CHARACTER	100
tooth_range_start	Tooth Number Range start (may be blank)	TINYINT	1
tooth_range_end	Tooth Number Range end (may be blank)	TINYINT	1
surface	Tooth Surface (may be blank)	CHARACTER	20
quadrant	Quadrant designation (may be blank)	CHARACTER	2
sextant	Sextant designation (may be blank)	CHARACTER	2
proc_date	Procedure Date	DATE	4
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
amt	Amount	MONEY	10

5.5 Patient's Treatment Planned Procedures

Stored Procedure Name: sp_getpatienttreatmentplan

Revision: 3

Inputs: Patient GUID (CHARACTER), BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the treatment planned procedures for a patient for a specified date range. The byCreateDate is set to 1 to retrieve the treatment planned procedures by entry date. This flag is set to 0 to retrieve the treatment planned procedures by the procedure date.

Example: call admin.sp_getpatienttreatmentplan('045e8d6e-a74d-4bda-a9ec-0db9609ee61a','01/01/2000','01/01/2010',0)

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4

patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
guar_last_name	Guarantor Last Name	CHARACTER	21
guar_first_name	Guarantor First Name	CHARACTER	16
proc_code	ADA Procedure Code	CHARACTER	10
description	Procedure Description	CHARACTER	100
tooth_range_start	Tooth Number Range start (may be blank)	TINYINT	1
tooth_range_end	Tooth Number Range end (may be blank)	TINYINT	1
surface	Tooth Surface (may be blank)	CHARACTER	20
quadrant	Quadrant designation (may be blank)	CHARACTER	2
sextant	Sextant designation (may be blank)	CHARACTER	2
proc_date	Procedure Date	DATE	4
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
amt	Amount	MONEY	10

5.6 Payment Types

View Name: v_payment_types
Revision: 1
Description: This view returns user-defined Payment Types from Dentrix.
Example: select * from admin.v_payment_types

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Payment Type Definition ID	INTEGER	4
descript	Payment Type Description	CHARACTER	52

5.7 Adjustment Types

View Name: v_adjustment_types

Revision: 1

Description: This view returns a list of user-defined Adjustment Types from Dentrix.

Example: select * from admin.v_adjustment_types

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Adjustment Type Definition ID	INTEGER	4
descript	Adjustment Type Description	CHARACTER	52

6. Notes

6.1 Notes

View Name: v_notes
Revision: 1
Description: This view returns various patient and other notes from Dentrix.
Patient Note Example: SELECT * FROM admin.v_notes WHERE notetype = 3 AND noteid = 10
Appointment Note Example: SELECT * FROM admin.v_notes WHERE notetype = 2 AND noteid = 232

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
notetype	Note Type: - 1=Transaction Note (procedures, payments, adjustments, etc.) - 2=Appointment Note - 3=Patient Note - 4=Procedure Code Progress Note - 5=Insurance Plan Note - 7=Dunning Message - 8=Billing Statement Message - 9=Continuing Care Motivational Note - 10=Payment Plan Note - 11=Insurance Claim Remarks for Unusual Services - 12=Perio Exam Miscellaneous Notes - 14=Walkout Statement Message - 16=Guarantor Account Note - 17=Guarantor Billing Statement Note - 22=Appointment Book Day Note - 26=Procedure Recommendation Note - 31=Post-Dated Check Note for Canada - 32=Payment Agreement Note - 33=Month End Wizard Setup - 34=Prescription Note - 35=Prescription "Sig" - 36=Medical Claim Information note for name and address of facility - 38=Patient Alert Note - 41=Document Center Document Note - 42=Electronic Claim Attachment note - 43=Referral Information Note - 44=Custom Lab Case Note	TINYINT	1

	- 45=Lab Case Note - 47=Lab Note - 49=Office Journal Entry Note - 51=Time Clock Employee Note - 54=Lab URL - 55=Tx Planner Case Note - 56=Tx Planner Case Note Template Text - 57=Proc Info for Tx Planner Consents - 58=Add'l Note for Tx Planner Consents - 59=Presenter items for Tx Plan Case - 60=Family Alert Note (for Patient Alerts) - 62=Custom Billing Statement Note - 63=Eligibility Status Note for Prim Ins - 65=Custom Procedure Note - 66=Patient Medical Alert Note - 100=Appointment Book Event Note		
Noteid	Unique ID relating to the note according to the Note Type (e.g., Appointment ID for Appointment Note, Patient ID for Patient Note, etc.).	TINYINT	1
notetext	Note Text	CHARACTER	8190

7. Office Journal

7.1 Patient's Office Journal Entries

Stored Procedure Name: sp_getpatientofficejournal
Revision: 2
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the Office Journal information for a patient.
Example: call admin.sp_getpatientofficejournal('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
contact_id	Office Journal ID	INTEGER	4
contact_date	Office Journal Date	DATE	4
contact_time	Office Journal Time (24 hour format)	TIME	4
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
contact_type	Office Journal Type: - 1=Letter - 2=Billing Statement - 3=Phone Call - 4=Reminder - 5=Miscellaneous - 6=Appointment – Purged - 7=Broken Appointment - 8=Archived Patient Appointment - 13=Message - 14=Referral Recap - 15=Referral Slip - 16=Referral Gratuity - 17=Web Referral - 18=Web Upload	TINYINT	1

	- 19=Web Communication - 20=HIPAA Privacy - 21=Web Eligibility Request - 22=Financing Requests - 23=Perio Letters		
contact_description	Office Journal Entry Description	CHARACTER	80
contact_note	Office Journal Entry Note	VARCHAR	5128
referral_id	Referral ID (will be 0 if this office journal entry is not for a referral)	INTEGER	4
ref_last_name	Referral Last Name (may be blank)	CHARACTER	32
ref_first_name	Referral First Name (may be blank)	CHARACTER	16

7.2 Provider's Office Journal Entries

Stored Procedure Name: sp_getproviderofficejournal
Revision: 2
Inputs: Provider ID (CHARACTER)
Throws: PROVIDER_NOT_FOUND
Description: This procedure returns the Office Journal information for a provider.
Example: call admin.sp_getproviderofficejournal('DDS1')

Column Name	Description	Type	Size
contact_id	Office Journal ID	INTEGER	4
contact_date	Office Journal Date	DATE	4
contact_time	Office Journal Time (24 hour format)	TIME	4
provider_id	Provider ID	CHARACTER	4
provider_last_name	Provider Last Name	CHARACTER	21
provider_first_name	Provider First Name	CHARACTER	16
contact_type	Office Journal Type: - 1=Letter - 2=Billing Statement - 3=Phone Call - 4=Reminder - 5=Miscellaneous - 6=Appointment – Purged - 7=Broken Appointment	TINYINT	1

	- 8=Archived Patient Appointment - 13=Message - 14=Referral Recap - 15=Referral Slip - 16=Referral Gratuity - 17=Web Referral - 18=Web Upload - 19=Web Communication - 20=HIPAA Privacy - 21=Web Eligibility Request - 22=Financing Requests - 23=Perio Letters		
contact_description	Office Journal Entry Description	CHARACTER	80
contact_note	Office Journal Entry Note	VARCHAR	5128

7.3 Referral's Office Journal Entries

Stored Procedure Name: sp_getreferralofficejournal
Revision: 1
Inputs: Referral ID (INTEGER)
Throws: REFERRAL_SOURCE_NOT_FOUND
Description: This procedure returns the Office Journal information for a Referred By/Other or Referred To referral source.
Example: call admin.sp_getreferralofficejournal(2)

Column Name	Description	Type	Size
contact_id	Office Journal ID	INTEGER	4
contact_date	Office Journal Date	DATE	4
contact_time	Office Journal Time (24 hour format)	TIME	4
contact_type	Office Journal Type: - 1=Letter - 2=Billing Statement - 3=Phone Call - 4=Reminder - 5=Miscellaneous - 6=Appointment – Purged - 7=Broken Appointment - 8=Archived Patient Appointment - 13=Message - 14=Referral Recap	TINYINT	1

	- 15=Referral Slip - 16=Referral Gratuity - 17=Web Referral - 18=Web Upload - 19=Web Communication - 20=HIPAA Privacy - 21=Web Eligibility Request - 22=Financing Requests - 23=Perio Letters		
contact_description	Office Journal Entry Description	CHARACTER	80
contact_note	Office Journal Entry Note	VARCHAR	5128
referral_id	Referral ID (will be 0 if this office journal entry is not for a referral)	INTEGER	4
ref_last_name	Referral Last Name (may be blank)	CHARACTER	32
ref_first_name	Referral First Name (may be blank)	CHARACTER	16

8. Patient and Account Information

8.1 Patient Demographics (View)

View Name: v_patient

Revision: 2

Description: This view retrieves all the records from the patient table.

Example: select * from admin.v_patient

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient last name	CHARACTER	21
first_name	Patient first name	CHARACTER	16
mi	Patient middle initial	CHARACTER	2
preferred_name	Patient preferred name	CHARACTER	16
salutation	Salutation	CHARACTER	32
title	Title	CHARACTER	10
guar_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
city	City	CHARACTER	26
state	State	CHARACTER	21
zipcode	ZIP Code	CHARACTER	15
home_phone	Patient home phone number	CHARACTER	17
work_phone	Patient work phone number	CHARACTER	17
work_ext	Patient work phone extension	CHARACTER	5
other_phone	Patient's other phone number	CHARACTER	17
mobile_phone	Patient's cell/pager number	CHARACTER	17
chart_num	Chart number	CHARACTER	20

social_sec_num	Patient Social Security Number	CHARACTER	10
pri_provider_id	Provider ID	CHARACTER	4
pri_provider_last_name	Primary Provider Last Name	CHARACTER	21
pri_provider_first_name	Primary Provider First Name	CHARACTER	16
sec_provider_id	Secondary Provider ID	CHARACTER	4
sec_provider_last_name	Secondary Provider Last Name	CHARACTER	21
sec_provider_first_name	Secondary Provider First Name	CHARACTER	16
gender	Patient's gender (1=Male; 2=Female)	TINYINT	1
status	Patient's status. - 1=Patient - 2=Non-Patient - 3=Inactive - 4=Archived	TINYINT	1
birth_date	Patient's birth date	DATE	4
first_visit_date	Patient's first visit date	DATE	4
last_visit_date	Patient's last visit date	DATE	4
last_missed_appointment	Last missed appointment date	DATE	4
num_of_missed_appointments	Number of missed appointments	SMALLINT	2
Email	Patient's e-mail address	CHARACTER	60
Fax	Fax Number	CHARACTER	17
Note	Patient Note	VARCHAR	5128
privacy_flags	Patient's Privacy Requests. Flags state of the following options for "Privacy Requests": - 0=no options are specified - 1="No phone calls" is specified - 2="No correspondence" (for Letters) is specified - 3="No phone calls" and "No correspondence" is specified - 4="Disclosure restrictions" is specified - 5="No phone calls" and "Disclosure restrictions" is specified - 6="No correspondence" and "Disclosure restrictions" is specified	TINYINT	1

	- 7=all three of the above options ("No phone calls", "No correspondence" and "Disclosure restrictions") is specified - 8="No E-mail" is specified - 10="No Text Message Correspondence" is specified - 20="No Postcard Correspondence" is specified - 63="No" has been specified for all communication methods (Phone calls, letters, disclosure, e-mail, text messages, postcards).		
employer_id	Employer ID	INTEGER	4
primdentalinsuredid	Primary Dental Insured ID	INTEGER	4
primmedicalinsuredid	Primary Medical Insured ID	INTEGER	4
secdentalinsuredid	Secondary Dental Insured ID	INTEGER	4
secmedicalinsuredid	Secondary Medical Insured ID	INTEGER	4
driverslicense	Driver's License number	CHARACTER	25

8.2 Patient Demographics (Stored Procedure)

Stored Procedure Name: sp_patient
Revision: 2
Description: This procedure retrieves all the records from the patient table.
[CAUTION: This query could run slowly depending on the number of patients.]
Example: call admin.sp_patient()

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient last name	CHARACTER	21
first_name	Patient first name	CHARACTER	16
mi	Patient middle initial	CHARACTER	2
preferred_name	Patient preferred name	CHARACTER	16
salutation	Salutation	CHARACTER	32
title	Title	CHARACTER	10

guar_id	Guarantor ID	INTEGER	4
guar_guid	Guarantor GUID	CHARACTER	36
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
city	City	CHARACTER	26
state	State	CHARACTER	21
zipcode	ZIP Code	CHARACTER	15
home_phone	Patient home phone number	CHARACTER	17
work_phone	Patient work phone number	CHARACTER	17
work_ext	Patient work phone extension	CHARACTER	5
other_phone	Patient's other phone number	CHARACTER	17
mobile_phone	Patient's cell/pager number	CHARACTER	17
chart_num	Chart number	CHARACTER	20
social_sec_num	Patient Social Security Number	CHARACTER	10
pri_provider_id	Provider ID	CHARACTER	4
pri_provider_last_name	Primary Provider Last Name	CHARACTER	21
pri_provider_first_name	Primary Provider First Name	CHARACTER	16
sec_provider_id	Secondary Provider ID	CHARACTER	4
sec_provider_last_name	Secondary Provider Last Name	CHARACTER	21
sec_provider_first_name	Secondary Provider First Name	CHARACTER	16
gender	Patient's gender (1=Male; 2=Female)	TINYINT	1
status	Patient's status. - 1=Patient - 2=Non-Patient - 3=Inactive - 4=Archived	TINYINT	1
birth_date	Patient's birth date	DATE	4
first_visit_date	Patient's first visit date	DATE	4
last_visit_date	Patient's last visit date	DATE	4
last_missed_appointment	Last missed appointment date	DATE	4

num_of_missed_appointments	Number of missed appointments	SMALLINT	2
email	Patient's e-mail address	CHARACTER	60
fax	Fax Number	CHARACTER	17
note	Patient Note	VARCHAR	5128
privacy_flags	Patient's Privacy Requests. Flags state of the following options for "Privacy Requests": - 0=no options are specified - 1="No phone calls" is specified - 2="No correspondence" (for Letters) is specified - 3="No phone calls" and "No correspondence" is specified - 4="Disclosure restrictions" is specified - 5="No phone calls" and "Disclosure restrictions" is specified - 6="No correspondence" and "Disclosure restrictions" is specified - 7=all three of the above options ("No phone calls", "No correspondence" and "Disclosure restrictions") is specified	TINYINT	1

8.3 Patient Clinical Notes

View Name: v_clinical_notes

Revision: 2

Description: This view returns patients' clinical notes entered in Dentrix.
[CAUTION: This query could run slowly depending on the number of patients]

Example: select * from admin.v_clinical_notes

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
cnotes_id	Clinical Notes ID	INTEGER	4
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient last name	CHARACTER	21
first_name	Patient first name	CHARACTER	16

mi	Patient middle initial	CHARACTER	2
provider_id	Provider ID	CHARACTER	4
create_date_time_stamp	Date and time the clinical note was created	TIMESTAMP	8
entry_date_time_stamp	Date and time the clinical note was locked	TIMESTAMP	8
note_text	Clinical Note text	VARCHAR	16384

8.4 Patient's Medical Alerts

Stored Procedure Name: sp_getpatientmedalerts
Revision: 1
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the medical alerts for a patient. A separate row will be returned for each medical alert that is entered for the patient.
Example: call admin.sp_getpatientmedalerts('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
def_id	Medical Alert Definition ID	INTEGER	4
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
med_alert	Medical Alert Description (will be blank if no medical alerts have been specified for the patient)	CHARACTER	52

8.5 Account / Guarantor Notes

View Name: v_account_notes
Revision: 1
Description: This view returns account / guarantor notes from Dentrix.
Example: select * from admin.v_account_notes

Column Name	Description	Type	Size

account_id	Guarantor ID	INTEGER	4
note_text	Guarantor Note	CHARACTER	3,998

8.6 Patient Address

View Name: v_address

Revision: 2

Description: This view returns addresses associated with patient and new patient records in Dentrix.

Example: select * from admin.v_address

Column Name	Description	Type	Size
address_addrid	Address ID	INTEGER	4
address_street1	Address line 1	CHARACTER	31
address_city	City	CHARACTER	26
address_ptrcount	Provider ID	TINYINT	1
address_state	State	CHARACTER	21
address_zipcode	ZIP Code	CHARACTER	15
address_phone	Home Phone	CHARACTER	17
address_street2	Address line 2	CHARACTER	31

9. Perio

9.1 Perio Exam Records

View Name: v_perio
Revision: 2
Description: This view retrieves patient ID, GUID and exam date for all perio exams in the database.
Example: select * from admin.v_perio

Column Name	Description	Type	Size
patid	Patient ID associated with perio exam	INTEGER	4
periodate	Perio exam date	DATE	4
patguid	Patient GUIID associated with perio exam	CHARACTER	36

9.2 Perio Exam Data

Stored Procedure Name: sp_getperioexam
Revision: 2
Inputs: Patient GUID (CHARACTER 36), Perio Exam Date (DATE)
Throws: PATIENT_NOT_FOUND
Description: This procedure returns the exam data for a specified perio exam for a patient.
Example: call admin.sp_getperioexam('045e8d6e-a74d-4bda-a9ec-0db9609ee61a','10/16/2014')

Column Name	Description	Type	Size
_exam	Perio Exam Information (includes Patient ID, Exam Date, Case Type and other elements saved for the perio exam). See **Table 1** for XML schema for the output for this column.	VARCHAR	8190
_quadrant1	Perio Exam Measurements (Quadrant 1). See **Table 2** for XML schema for the output for this column.	VARCHAR	8190
_quadrant2	Perio Exam Measurements (Quadrant 2). See **Table 2** for XML schema for the output for this column.	VARCHAR	8190
_quadrant3	Perio Exam Measurements (Quadrant 3). See **Table 2** for XML schema for the output for this column.	VARCHAR	8190

_quadrant4	Perio Exam Measurements (Quadrant 4). See **Table 2** for XML schema for the output for this column.	VARCHAR	8190

9.2.1 Perio Exam Data XML Schema

This section provides details for the XML schema returned using the sp_getperioexam stored procedure described in this document. **Table 1** below describes the schema for the **_exam** column and **Table 2** describes the schema for the **_quadrant1**, **_quadrant2**, **_quandrant3** and **_quadrant4** columns.

Table 1 – Example showing XML schema used for output for the '_exam' column for sp_getperioexam

```
<?xml version="1.0" encoding="UTF-8"?>
<PerioExam>
   <PatId>3</PatId>
   <ExamDate>2015-02-27</ExamDate>
   <ProvId>DDS1</ProvId>
   <CaseType>0</CaseType>
   <PatStatus>0</PatStatus>
   <PerioStatus>0</PerioStatus>
   <ExamNotes>
      <Gingiva>
         <Color>0</Color>
         <Texture>0</Texture>
         <Margins>0</Margins>
         <Attachment>0</Attachment>
         <Papillae>0</Papillae>
         <Sulcus>0</Sulcus>
         <Contour>0</Contour>
         <Bleeding>0</Bleeding>
         <Suppuration>0</Suppuration>
      </Gingiva>
      <X-rays>
         <Bone_Loss>0</Bone_Loss>
         <Bone_Defects>0</Bone_Defects>
      </X-rays>
      <OralHygiene>
         <Plaque>0</Plaque>
         <Calculus>0</Calculus>
         <Stain>0</Stain>
      </OralHygiene>
   </ExamNotes>
   <History>false</History>
</PerioExam>
```

Table 2 – Example showing XML schema used for output for the '_quadrant1', '_quadrant2', '_quadrant3' and '_quadrant4' columns for sp_getperioexam (showing standard fields / elements relating to this procedure). The example below shows the XML returned for Quadrant 1 of a patient's period exam data.

```
<?xml version="1.0" encoding="UTF-8"?>
<Quadrant1>
   <Tooth>
      <ToothNum>1</ToothNum>
      <Facial>
```

```
            <PD>1,2,3</PD>
            <GM>-1,-1,-1</GM>
            <CAL>1,2,3</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,0,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
         </Facial>
         <Lingual>
            <PD>-1,-1,-1</PD>
            <GM>-1,-1,-1</GM>
            <CAL>-1,-1,-1</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>0,-2,0</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
         </Lingual>
         <Plaque>0</Plaque>
         <Mobility>0</Mobility>
         <BoneLoss>0</BoneLoss>
         <ToothState>0</ToothState>
      </Tooth>
      <Tooth>
         <ToothNum>2</ToothNum>
         <Facial>
            <PD>2,1,3</PD>
            <GM>-1,-1,-1</GM>
            <CAL>2,1,3</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,0,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
         </Facial>
         <Lingual>
            <PD>-1,-1,-1</PD>
            <GM>-1,-1,-1</GM>
            <CAL>-1,-1,-1</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>0,-2,0</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
         </Lingual>
         <Plaque>0</Plaque>
         <Mobility>0</Mobility>
         <BoneLoss>0</BoneLoss>
         <ToothState>0</ToothState>
      </Tooth>
      <Tooth>
         <ToothNum>3</ToothNum>
         <Facial>
            <PD>1,2,3</PD>
            <GM>-1,-1,-1</GM>
            <CAL>1,2,3</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,0,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
         </Facial>
      <Lingual>
         <PD>-1,-1,-1</PD>
         <GM>-1,-1,-1</GM>
         <CAL>-1,-1,-1</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>0,-2,0</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Lingual>
      <Plaque>0</Plaque>
      <Mobility>0</Mobility>
      <BoneLoss>0</BoneLoss>
      <ToothState>0</ToothState>
   </Tooth>
   <Tooth>
      <ToothNum>4</ToothNum>
      <Facial>
         <PD>2,1,3</PD>
         <GM>-1,-1,-1</GM>
         <CAL>2,1,3</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>-2,-2,-2</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Facial>
      <Lingual>
         <PD>-1,-1,-1</PD>
         <GM>-1,-1,-1</GM>
         <CAL>-1,-1,-1</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>-2,-2,-2</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Lingual>
      <Plaque>0</Plaque>
      <Mobility>0</Mobility>
      <BoneLoss>0</BoneLoss>
      <ToothState>0</ToothState>
   </Tooth>
   <Tooth>
      <ToothNum>5</ToothNum>
      <Facial>
         <PD>3,2,4</PD>
         <GM>-1,-1,-1</GM>
         <CAL>3,2,4</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>0,-2,0</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Facial>
      <Lingual>
         <PD>-1,-1,-1</PD>
         <GM>-1,-1,-1</GM>
         <CAL>-1,-1,-1</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>0,-2,0</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
```

```
        </Lingual>
        <Plaque>0</Plaque>
        <Mobility>0</Mobility>
        <BoneLoss>0</BoneLoss>
        <ToothState>0</ToothState>
    </Tooth>
    <Tooth>
        <ToothNum>6</ToothNum>
        <Facial>
            <PD>3,1,2</PD>
            <GM>-1,-1,-1</GM>
            <CAL>3,1,2</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,-2,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
        </Facial>
        <Lingual>
            <PD>-1,-1,-1</PD>
            <GM>-1,-1,-1</GM>
            <CAL>-1,-1,-1</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,-2,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
        </Lingual>
        <Plaque>0</Plaque>
        <Mobility>0</Mobility>
        <BoneLoss>0</BoneLoss>
        <ToothState>0</ToothState>
    </Tooth>
    <Tooth>
        <ToothNum>7</ToothNum>
        <Facial>
            <PD>4,3,1</PD>
            <GM>-1,-1,-1</GM>
            <CAL>4,3,1</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,-2,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
        </Facial>
        <Lingual>
            <PD>-1,-1,-1</PD>
            <GM>-1,-1,-1</GM>
            <CAL>-1,-1,-1</CAL>
            <MGJ>-1,-1,-1</MGJ>
            <FG>-2,-2,-2</FG>
            <Bleeding>0,0,0</Bleeding>
            <Suppuration>0,0,0</Suppuration>
        </Lingual>
        <Plaque>0</Plaque>
        <Mobility>0</Mobility>
        <BoneLoss>0</BoneLoss>
        <ToothState>0</ToothState>
    </Tooth>
    <Tooth>
        <ToothNum>8</ToothNum>
```

```
      <Facial>
         <PD>3,2,4</PD>
         <GM>-1,-1,-1</GM>
         <CAL>3,2,4</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>-2,-2,-2</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Facial>
      <Lingual>
         <PD>-1,-1,-1</PD>
         <GM>-1,-1,-1</GM>
         <CAL>-1,-1,-1</CAL>
         <MGJ>-1,-1,-1</MGJ>
         <FG>-2,-2,-2</FG>
         <Bleeding>0,0,0</Bleeding>
         <Suppuration>0,0,0</Suppuration>
      </Lingual>
      <Plaque>0</Plaque>
      <Mobility>0</Mobility>
      <BoneLoss>0</BoneLoss>
      <ToothState>0</ToothState>
   </Tooth>
</Quadrant1>
```

9.2.2 XML Elements for sp_getperioexam '_exam' column

This section provides details for the elements used in the XML output returned from the sp_getperioexam stored procedure described in this document.

Table 3

Element Name	Description
`<PerioExam>`	This is the root element used to mark the beginning and ending of the perio exam information returned for the _exam column.
`<PatId>`	Used to indicate the Patient ID of the patient corresponding to the perio exam. The Patient ID will be specified as an integer.
`<ExamDate>`	Used to indicate the Date of the perio exam. The perio exam date will be specified in the format YYYY-MM-DD.
`<ProvId>`	Used to indicate the Provider ID of the provider corresponding to the perio exam.
`<CaseType>`	Used to indicate the value for the case type specified for the perio exam as follows: 0=None 1=Type I-Gingival Diseases

	2=Type II-Slight Periodontitis (1989) 3=Type III-Moderate Periodontitis (1989) 4=Type IV-Advanced Periodontitis (1989) 5=Type V-Refractory Progressive (1989) 6=Type II-Chronic Periodontitis 7=Type III-Aggressive Periodontitis 8=Type IV-Periodontitis as a Manifestation of Systemic Diseases 9=Type V-Necrotizing Periodontal Diseases 10=Type VI-Abscesses of the Periodontium 11=Type VII-Periodontitis Associated With Endodontic Lesions 12=Type VIII-Developmental or Acquired Deformities and Conditions
`<PatStatus>`	Used to indicate the status of the patient corresponding to the perio exam: 0=None 1=Adult 2=Juvenile 3=Adult, Juvenile 4=Pregnant 5=Adult, Pregnant 6=Juvenile, Pregnant
`<PerioStatus>`	Used to indicate the status of the patient corresponding to the perio exam: 0=None 1=Chronic 2=Acute 3=Chronic, Acute
`<ExamNotes>`	Used to mark the beginning and ending of the perio notes information, such as indicators for Gingiva, Oral Hygiene and X-rays.
`<Gingiva>`	Used to mark the beginning and ending of the indicators for the patient's Gingiva corresponding to the perio exam.
`<Color>`	Used to indicate the color of the patient's gingiva. The possible values for this element are as follows: 0=Coral Pink 1=Pink 2=Red 3=Magenta 4=Pink with Magenta areas 5=Light 6=Dark

`<Texture>`	Used to describe the texture of the patient's gingiva. The possible values for this element are as follows: 0=Stippled 1=Glossy 2=Rough
`<Margins>`	Used to describe the margins of the patient's gingiva. The possible values for this element are as follows: 0=Knife Edged 1=Clefts 2=Cuffing
`<Attachment>`	Used to describe the attachment of the patient's gingiva. The possible values for this element are as follows: 0=Normal 1=Recession 2=Enlarged Tissue
`<Papillae>`	Used to describe the papillae of the patient's gingiva. The possible values for this element are as follows: 0=Scalloped 1=Blunted 2=Enlarged 3=Punched Out 4=Cratered
`<Sulcus>`	Used to describe the sulcus of the patient's gingiva. The possible values for this element are as follows: 0=None 1=Blood 2=Exudates
`<Contour>`	Used to describe the contour of the patient's gingiva. The possible values for this element are as follows: 0=Normal 1=Irregular 2=Recessions
`<Bleeding>`	Used to describe the bleeding level of the patient's gingiva. The possible values for this element are as follows: 0=None 1=Mild

	2=Moderate 3=Severe
`<Suppuration>`	Used to describe the suppuration level of the patient's gingiva. The possible values for this element are as follows: 0=None 1=Mild 2=Moderate 3=Severe
`<X-rays>`	Used to mark the beginning and ending of the indicators for the patient's X-rays corresponding to the perio exam.
`<OralHygiene>`	Used to mark the beginning and ending of the indicators for the patient's Oral Hygiene corresponding to the perio exam.
`<Plaque>`	Used to indicate the plaque level for the patient's oral hygiene. The possible values for this element are as follows: 0=None 1=Light 2=Moderate 3=Heavy
`<Calculus>`	Used to indicate the calculus level for the patient's oral hygiene. The possible values for this element are as follows: 0=None 1=Light 2=Moderate 3=Heavy
`<Stain>`	Used to indicate the stain level for the patient's oral hygiene. The possible values for this element are as follows: 0=None 1=Light 2=Moderate 3=Heavy
`<bone_loss>`	Used to indicate the bone less level for the patient's x-rays. The possible values for this element are as follows: 0=None 1=Mild 2=Moderate 3=Severe

`<bone_defects>`	Used to indicate the bone defects level for the patient's x-rays. The possible values for this element are as follows: 0=None 1=Craters 2=Wells 3=Furcations
`<History>`	Used to indicate if the perio exam has been moved to history. True=Perio exam has been moved to history False=Perio exam has not been moved to history

9.2.3 XML Elements for sp_getperioexam '_exam' column

This section provides details for the elements used in the XML output returned from the sp_getperioexam stored procedure described in this document.

Table 4

Element Name	Description
`<Quadrant>`	This is the root element used to mark the beginning and ending of the perio exam data returned for the **_quadrant1**, **_quadtrant2**, **_quadrant3** and **_quadrant4** columns.
`<Tooth>`	Used to mark the beginning and ending of the perio exam data for a given tooth.
`<ToothNum>`	Used to indicate the tooth number.
`<Facial>`	Used to mark the beginning and ending of the perio exam measurements for the facial side of a tooth.
`<Facial>`	Used to mark the beginning and ending of the perio exam measurements for the facial side of a tooth.
`<Lingual>`	Used to mark the beginning and ending of the perio exam measurements for the lingual side of a tooth.
`<PD>`	Used to indicate the probing depth measurement for the lingual or facial side of a tooth, with a possible measurement value of 0 to 9 for the Left (Distal), Center and Right (Mesial) surfaces of the tooth. If no measurement is entered, -1 will be specified for a given surface for this element.
`<GM>`	Used to indicate the gingival margin measurement for the lingual or facial side of a tooth, with a possible measurement value of 0 to 9 for the Left (Distal), Center and Right (Mesial) surfaces of the tooth. If no

	measurement is entered, -1 will be specified for a given surface for this element.
`<CAL>`	Used to indicate the clinical attachment level measurement for the lingual or facial side of a tooth, with a possible measurement value of 0 to 9 for the Left (Distal), Center and Right (Mesial) surfaces of the tooth. If no measurement is entered, -1 will be specified for a given surface for this element.
`<MGJ>`	Used to indicate the mucco gingival junction measurement for the lingual or facial side of a tooth, with a possible measurement value of 0 to 9 for the Left (Distal), Center and Right (Mesial) surfaces of the tooth. If no measurement is entered, -1 will be specified for a given surface for this element.
`<FG>`	Used to indicate the furcation grade measurement for the lingual or facial side of a tooth, with the following possible measurement value for the Left (Distal), Center and Right (Mesial) surfaces of the tooth. -2= Blank 0= F0: (none) 1= F1: Probe root indentation 2= F2: Penetrates into furcation 3= F3: Through furcation – soft tissue 4= F4: Furcation open void of soft tissue
`<Bleeding>`	Used to indicate the bleeding level measurement for the lingual or facial side of a tooth, with a possible value of 0 to 9. If no measurement is entered, -1 will be specified for this element.
`<Suppuration>`	Used to indicate the suppuration level measurement for the lingual or facial side of a tooth, with a possible value of 0 to 9. If no measurement is entered, -1 will be specified for this element.
`<Plaque>`	Used to the plaque level for a given tooth. The possible values for this element are as follows: 0=None 1=Light 2=Moderate 3=Heavy
`<Mobility>`	Used to indicate the mobility measurement for a given tooth, with a possible value of 0 to 4.
`<BoneLess>`	Used to indicate the bone loss level for a given tooth. The possible values for this element are as follows: 0=Normal 1=Mild

	2=Moderate 3=Severe
`<ToothState>`	Used to indicate the state of the tooth for the following conditions (if applicable): Crown Impacted – distal Impacted – mesial Implant Implant & Crown Missing Pontic Unerupted None

10. Practice Definitions (Misc.)

10.1 Billing Types

View Name: v_billing_types
Revision: 1
Description: This view returns a list of user-defined Billing Types from Dentrix.
Example: select * from admin.v_billing_types

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Billing Type Definition ID	INTEGER	4
descript	Billing Type Description	CHARACTER	52

10.2 Document Types

View Name: v_document_types
Revision: 2
Description: This view returns a list of Document Types used in the Document Center.
Example: select * from admin.v_document_types

Column Name	Description	Type	Size
dc_doctype_id	Document Type ID	INTEGER	4
doctype_desc	Document Type Description	CHARACTER	41

10.3 Practice Medical Alerts

View Name: v_practice_medical_alerts
Revision: 1
Description: This view returns user-defined Medical Alerts from Dentrix.
Example: select * from admin.v_practice_medical_alerts

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
def_id	Medical Alert Definition ID	INTEGER	4
descript	Medical Alert Description	CHARACTER	52

11. Practice Setup

11.1 Practice

View Name: v_practice_information
Revision: 1
Description: This view returns information for the practice.
Example: select * from admin.v_practice_information

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
practice_name	Practice Name	CHARACTER	60
address_line1	Address Line 1	CHARACTER	31
address_line2	Address Line 2	CHARACTER	31
City	City	CHARACTER	26
State	State	CHARACTER	16
zip_code	Zip Code	CHARACTER	15
phone	Phone	CHARACTER	17
phone_ext	Phone Extension	CHARACTER	5
deposit_slip	Bank Deposit Number for the Practice	CHARACTER	30
fiscal_beg_month	Fiscal year's beginning month (1-12)	TINYINT	1
statement_info	Statement Info flag. Designates the information to be used for billing statements: - 0=Practice information is to be used - 1=Provider information is to be used (guarantor's Prov1)	TINYINT	1

11.2 Procedure Codes

View Name: v_proccodes
Revision: 1
Description: This view returns procedure codes that have been setup for the practice.
Example: select * from admin.v_proccodes

Column Name	Description	Type	Size
automodifiedtimestamp	Modified Date/Time Stamp	TIMESTAMP	8
proccodeid	Linked from Appointments records, Fee Schedule records, Patient Education Topic Links records, Procedure Log records. Also links to Notes record (NoteType of 4 or 26) for procedure code progress note and recommendation note.	INTEGER	4
adacode	Procedure Code / ADA CDT Code	CHARACTER	10
abbrevdescript	Abbreviated description. Used for appointment reason for appointments. May be blank.	CHARACTER	10
descript	Procedure description	CHARACTER	100
alt1	An alternate code. Defaults to be called "Code 3" but may be named by the user. May be blank.	CHARACTER	10
alt2	An alternate code. Defaults to be called "Code 4" but may be named by the user. See alt1.	CHARACTER	10
alt3	An alternate code. Defaults to be called "Code 5" but may be named by the user. See alt1.	CHARACTER	10
category	Category for the procedure code or diagnostic code: - If this is a procedure code (diagcode is 0), links to a Definitions record (DefType of 16) - If this is a diagnostic code (DiagCode is 1), links to a Definitions record (DefType of 24) This may be 0. If this is a multi-code, this field will be 999.	SMALLINT	2
multicode	Enabled flag for record if it is for a multi-code: - 0= a multi-code that is disabled (not available for use) - 1= a multi-code that is enabled	BIT	1

multicodetype	Multi-code type for record if it is for a multi-code: - 0= “Standard” - 1=“Bridge”	TINYINT	1
nestmulticode	Nested flag for record if it is for a multi-code: - 0= the multi-code does not include any other multi-codes - 1= the multi-code includes one or more multi-codes as procedures for the multi-code	TINYINT	1
startcompdatereq	Flag for start and completed dates required for procedure: - 0= date range not required, only completed date - 1 start and completed dates are required	TINYINT	1
condition	Condition flag: - 0= “Condition” checkbox is not marked - 1= “Condition” checkbox is marked Conditions are used differently than other procedure codes.	TINYINT	1
medcrosscode	“Flag for Medical Cross Coding” flag: - 0= checkbox is not marked - 1= checkbox is marked—when a procedure is posted with this code the user will be able to add medical billing information	TINYINT	1
donotbillinsflag	“Do Not Bill to Dental Insurance” flag: - 0= checkbox is not marked - 1= checkbox is marked—procedures posted with this code will not be included when insurance claims are created	BIT	1
diagflag	Dental diagnostic cross code flag: - 0= the procedure code is not cross coded with diagnostic codes or that none are marked to attach automatically when the procedure code is posted - 1= the procedure code has one or more cross coded diagnostic codes that are	TINYINT	1

	marked to attach automatically when the procedure code is posted		
diagcode	Dental diagnostic code flag: - 0= not a dental diagnostic code (it is a procedure code or multi-code) - 1= dental diagnostic code	TINYINT	1
diagnotetoclin	Controls how the diagnostic code progress note (when there is progress note text) is used when a procedure that has the diagnostic code attached is posted. Also see notetoclin column. - 0= progress note will not be used - 1= progress note will be appended to the procedure’s progress note - 2= progress note will be appended to the procedure’s entry in the clinical notes for the patient	TINYINT	1
printnote	Controls how the recommendation note for the procedure code is used for a walkout statement that includes the procedure: - 0= “Print Note on Walkout” checkbox is not marked - 1= “Print Note on Walkout” checkbox is marked	TINYINT	1
notetoclin	Controls how the procedure code progress note or diagnostic code progress note (when there is progress note text) is used when the procedure code is completed, the condition is posted, or the diagnostic code is posted to a patient (not attached to a procedure): - 0= progress note will not be used - 1= progress note will be copied to the procedure/condition/diagnostic progress note - 2= progress note will be copied to the clinical notes for the patient	TINYINT	1
labexpense	If ChartStatus is 90 (for Ledger) and ProcLogClass is 0 for a balance forward or initial balance: The amount for over 90. May be 0.00.	MONEY	10

matexpense	If ChartStatus is 90 (for Ledger) and ProcLogClass is 0 for a balance forward or initial balance: The amount for 60-90. May be 0.00.	MONEY	10
noteid	Links to Notes record (see v_notes where notetype = 4) for procedure code progress note and Notes record (see v_notes where notetype = 26) procedure code recommendation note. May be 0. This is the same as ProcCodeId. Note: Recommendation note text may include document file names that were selected as "Recommendation Documents", enclosed in angle brackets (e.g. <<extractions.doc>>).	INTEGER	4
treatmentarea	Indicates treatment area for procedure code: - 0= "Surface" is selected - 1="Tooth" is selected - 2= "Mouth" is selected - 3= "Root" is selected - 4= "Quadrant" is selected - 5= "Sextant" is selected - 6= "Other" (not used) - 7= "Arch" is selected – new for G5.1	TINYINT	1
removetooth	Controls display of tooth for procedure codes for extractions and primary/permanent changes: - 0= "Remove Tooth" checkbox is not marked—a tooth is not removed when the procedure code is used for the tooth - 1= "Remove Tooth" checkbox is marked—a tooth with this procedure code will be removed (will not be shown)* - 2= "Remove Tooth" checkbox is marked and grayed—a tooth with this procedure code will be displayed as a primary tooth (this setting can only be saved for procedure code 15010) *This setting only acts upon completed procedures and conditions/ diagnoses, not on treatment planned procedures.	TINYINT	1
rangestart	Tooth number or beginning of tooth number range for a procedure. May be	TINYINT	1

	blank. See 'Dentrix Tooth Numbering.pdf' in the 'Misc Docs' folder in *DentrixSDK_2.1.zip* for how tooth numbers are stored.		
rangeend	Ending of tooth number range for a procedure. May be blank.	TINYINT	1
chartshow	"Show in Chart" flag: - 0= "Show in Chart" checkbox is not marked - 1= "Show in Chart" checkbox is marked—the procedure code will display for its category in the "Procedure Codes" panel of the Chart without clicking "<<More Codes>>"	BIT	1
addtlcodesflag	Flags how to map to the correct code to post when the procedure code is used, according to treatment area and number of surfaces, roots or times the procedure code is posted. If "Surface" is selected for treatment area (0): - 0= "None" option is selected for Surface Flag - 1= "Procedure Codes for Additional Surfaces" is selected for Surface Flag If "Tooth" is selected for treatment area (1): - 0= "None" option is selected for Procedure Flag - 2= "Procedure Code for Each Additional" option is selected for Procedure Flag If "Mouth" is selected for treatment area (2): - 0= "None" option is selected for Mouth Flag - 2= "Procedure Number for Each Additional" option is selected for Mouth Flag - 3= "Use Default Teeth for Range" option is selected for Mouth Flag - 4= "Use Selected Teeth for Range" option is selected for Mouth Flag If "Root" is selected for treatment area (3):	TINYINT	1

	- 0= "None" option is selected for Root Flags dialog box - 1= "Procedure Code for Additional Roots" option is selected for Root Flags dialog box If "Quadrant" or "Sextant" is selected for treatment area (4 or 5), this flag is not used.		
mapcode1	Mapped code to use if AddtlCodesFlag is 1 or 2. See AddtlCodesFlag. Links to Procedure Codes record (see adacode column in this table view). - If the treatment area is "Surface" or "Root", this is the first of up to 5 additional procedure codes to map to according to the number of surfaces or roots for the tooth - If the treatment area is "Mouth" or "Tooth", this is the procedure code to map to if the procedure is subsequently posted after the first time for the same patient for the same date. May be blank.	CHARACTER	10
mapcode2	Mapped code to use if AddtlCodesFlag is 1 or 2 (see AddtlCodesFlag) and the treatment area is "Surface" or "Root". See MapCode1.	CHARACTER	10
mapcode3	Mapped code to use if AddtlCodesFlag is 1 or 2 (see AddtlCodesFlag) and the treatment area is "Surface" or "Root". See MapCode1.	CHARACTER	10
mapcode4	Mapped code to use if AddtlCodesFlag is 1 or 2 (see AddtlCodesFlag) and the treatment area is "Surface" or "Root". See MapCode1.	CHARACTER	10
mapcode5	Mapped code to use if AddtlCodesFlag is 1 or 2 (see AddtlCodesFlag) and the treatment area is "Surface" or "Root". See MapCode1.	CHARACTER	10

bicuspidcode	Not used for Dentrix U.S. version. Used only for bicuspid codes for Canada.	CHARACTER	10
antcode	Links to Procedure Codes record. - The anterior procedure code for "Procedure Codes for Alternate Treatment Areas" if "Surface" or "Tooth" is selected for treatment area and the option is used. - The maxillary procedure code for "Procedure Codes for Alternate Arch" if "Mouth" is selected for treatment area and the option is used.	CHARACTER	10
postcode	Links to Procedure Codes record. - The posterior procedure code for "Procedure Codes for Alternate Treatment Areas" if "Surface" or "Tooth" is selected for treatment area and the option is used. - The mandibular procedure code for "Procedure Codes for Alternate Arch" if "Mouth" is selected for treatment area and the option is used. May be blank.	CHARACTER	10
primcode	Links to Procedure Codes record. The primary procedure code for "Procedure Codes for Alternate Treatment Areas" if "Surface" or "Tooth" is selected for treatment area and the option is used. May be blank.	CHARACTER	10
permcode	Links to Procedure Codes record. The permanent procedure code for "Procedure Codes for Alternate Treatment Areas" if "Surface" or "Tooth" is selected for treatment area and the option is used. May be blank.	CHARACTER	10
multinumofcodes	Total number of procedure codes within the multi-code if the record is for a multi-code. There is a maximum of 8 procedure codes within a multi-code.	SMALLINT	2

multicode1id	First procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode2id	Second procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode3id	Third procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode4id	Fourth procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode5id	Fifth procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode6id	Sixth procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode7id	Seventh procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
multicode8id	Eighth procedure code for the multi-code if the record is for a multi-code. Links to Procedure Codes record. May be blank.	INTEGER	4
treatflag	Flags how to map to the correct code to post when the procedure code is used, according to treatment area and dentition/position or arch. If “Surface” is selected for treatment area (0): - 0= “None” option is selected for Dentition/Position Flags - 1= “Procedure Codes for Alternate Treatment Areas” is selected for Dentition/Position Flags If “Tooth” is selected for treatment area (1): - 0= “None” option is selected for Dentition/Position Flags	TINYINT	1

	- 2= "Procedure Code for Alternate Treatment Areas" option is selected for Dentition/Position Flags If "Mouth" is selected for treatment area (2): - 0= "None" option is selected for Arch Flags - 1= "Procedure Codes for Alternate Arch" option is selected for Arch Flags If "Root", "Quadrant" or "Sextant" is selected for treatment area (3, 4 or 5), this flag is not used.		
autorecall	Links to Continuing Care Types record (see recallid in v_recall_type table view) for auto continuing care to generate a new or update and existing Continuing Care Pending record for the patient and continuing care type when this procedure code is posted as completed.	SMALLINT	2
appttye	Appointment Type. Links to Appointment Types table view (see def_id in v_appointment_types). May be blank. Used for procedure codes and multi codes. This is the default "Appointment Type" for an appointment when the procedure code is selected for the appointment reason. May be 0.	TINYINT	1
minlen	Procedure time. The "Total Time Units" from the Appointment Time Pattern dialog box. Used for procedure codes and multi codes. This is the default number of time units (appointment increments of 5, 10, 15, 20, 30 minutes) scheduled by default when the procedure code is selected for the appointment reason. May be 0.	SMALLINT	2
followup	Bits to indicate the state of the "Appointment Check List" checkboxes for the appointment. There may be 0 to 12 checkboxes. Each bit is a Foreign Key that links to a record in the Appointment Check	TINYINT	1

	List table view for the appointment check list items. May be 0.

12. Prescriptions

12.1 Practice Prescriptions

View Name: v_rxlist
Revision: 1
Description: This view retrieves all prescriptions defined for the practice.
Example: select * from admin.v_rxlist

Column Name	Description	Type	Size
rx_definition_id	Prescription definition ID	INTEGER	4
drug_name	Drug name	CHARACTER	50
description	Drug description	CHARACTER	50
rx_date	Prescription date	DATE	4
active	Active Flag. Flags whether or not this prescription can be selected for other patients: - 0=prescription is not available from Prescriptions Setup or for selection for other patients (it exists because it is linked from one or more Prescriptions for Patients records) - 1=prescription is available from Prescriptions Setup and for selection for other patients	TINYINT	1
std	Standard Flag. Flags whether or not this prescription is a "Standard" prescription: - 0=prescription was selected for a patient and edited at that time for that patient prescription - 1=prescription was set up from Prescriptions Setup and/or it is a patient prescription that was not edited when selected for the patient (it may have been edited from Prescriptions Setup afterwards)	TINYINT	1
dispense	Dispense	CHARACTER	50
refills	Refills	INTEGER	4
sig	Directions (sig/signa)	VARCHAR	5128
aswritten	As written flag. Flags the following: - 0=Dispense as written - 1=Generic Substitution Permitted	TINYINT	1

note	Patient instruction	VARCHAR	5128

12.2 Patient Prescriptions

View Name: v_rx_patient

Revision: 1

Description: This view retrieves all the prescriptions given for all patients. The rxdefid is the ID that helps determine which record to retrieve from rxlist_view.

Example: select * from admin.v_rx_patient

Column Name	Description	Type	Size
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient last name	CHARACTER	21
first_name	Patient first name	CHARACTER	16
mi	Patient middle initial	CHARACTER	2
rx_id	Prescription ID	INTEGER	4
rx_definition_id	Prescription definition ID	INTEGER	4
rx_date	Prescription date	DATE	4
provider_id	Provider ID	CHARACTER	5
drug_name	Drug name	CHARACTER	50
description	Drug description	CHARACTER	50
dispense	Dispense	CHARACTER	50
Refills	Refills	INTEGER	4
Sig	Directions (sig/signa)	VARCHAR	5128
aswritten	Dispense as written	TINYINT	1
note	Patient instruction	VARCHAR	5128

13. Provider Information

13.1 Providers (Stored Procedure)

Stored Procedure Name: sp_getallproviders
Revision: 2
Inputs: None
Description: This procedure retrieves provider information.
Example: call admin.sp_getallproviders()

Column Name	Description	Type	Size
provider_id	Provider ID	CHARACTER	4
last_name	Provider last name	CHARACTER	21
first_name	Provider first name	CHARACTER	16
mi	Provider middle initial	CHARACTER	2
Suffix	Provider suffix (MD, DDS, DMD, etc)	CHARACTER	21
work_phone	Provider work phone number	CHARACTER	17
work_phone_ext	Provider work phone extension	CHARACTER	5
tin	Provider TIN number	CHARACTER	16
npi	Provider NPI number	CHARACTER	16
provider_class	Provider Class (0=Primary; 1=Secondary)	CHARACTER	4
provider_specialty	Provider Specialty Description	CHARACTER	52
provider_color	Provider Color (RGB value returned in HEX format)	BINARY	3

13.2 Providers (View)

View Name: v_provider
Revision: 2
Description: This view retrieves provider information.
Example: select * from admin.v_provider

Column Name	**Description**	**Type**	**Size**
provider_id	Provider ID	CHARACTER	4

last_name	Provider last name	CHARACTER	21
first_name	Provider first name	CHARACTER	16
Mi	Provider middle initial	CHARACTER	2
Suffix	Provider suffix (MD, DDS, DMD, etc)	CHARACTER	21
work_phone	Provider work phone number	CHARACTER	17
work_phone_ext	Provider work phone extension	CHARACTER	5
Tin	Provider TIN number	CHARACTER	16
Npi	Provider NPI number	CHARACTER	16
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
City	City	CHARACTER	26
State	State	CHARACTER	21
zip_code	ZIP Code	CHARACTER	15
Ssn	Provider Social Security Number	CHARACTER	10
state_id	Provider State ID	CHARACTER	16
medicaid_id	Provider Medicaid ID	CHARACTER	16
drug_id	Provider Drub ID	CHARACTER	16
issecondaryprovider	Secondary Provider Flag	TINYINT	1
specialty_id	Provider Specialty ID	TINYINT	1
bluecross_id	Provider Blue Cross ID	CHARACTER	16
isnonperson	Non-person Flag	BIT	1
isblueshield	Blue Shield flag	TINYINT	1
Inactive	Inactive Flag (0=Active; 1=Inactive)	BIT	1
prov_num	Provider #	CHARACTER	16
other_id	Other ID	CHARACTER	16

13.3 Provider's Production

Stored Procedure Name: sp_getproviderproduction

Revision: 2

Inputs: provider(CHAR 4) BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the provider production including finance charges and late charges for a given date range. The byCreateDate is set to 1 to retrieve the production by entry date. This flag is set to 0 to retrieve the production by the procedure date.

Example: call admin.sp_getproviderproduction('DDS1','01/01/2000','01/01/2010',0)

Column Name	Description	Type	Size
provider_id	Provider ID	VARCHAR	4
last_name	Provider Last Name	CHARACTER	21
first_name	Provider First Name	CHARACTER	16
production	Provider Production	MONEY	10
Tin	Provider TIN number	CHARACTER	16
Npi	Provider NPI number	CHARACTER	26

13.4 All Providers' Production

Stored Procedure Name: sp_getallprovidersproduction

Revision: 2

Inputs: BeginDate (DATE), EndDate (DATE), byCreateDate (SMALLINT)

Description: This procedure returns the provider production for <u>all</u> providers, including finance charges and late charges for a given date range. The byCreateDate is set to 1 to retrieve the production by entry date. This flag is set to 0 to retrieve production by the procedure date.
[CAUTION: This query could run slow depending on the number of providers]

Example: call admin.sp_getallprovidersproduction('01/01/2000','01/01/2010',0)

Column Name	Description	Type	Size
provider_id	Provider ID	VARCHAR	4
last_name	Provider Last Name	CHARACTER	21
first_name	Provider First Name	CHARACTER	16
production	Provider Production	MONEY	10
Tin	Provider Registration ID	INTEGER	4
Npi	NPI	INTEGER	4

13.5 Provider Analysis Totals

Stored Procedure Name: sp_getprovidermonthtotals
Revision: 3
Inputs: Provider ID (CHARACTER), Month (MONTH), Year (YEAR)
Required Inputs: All
Throws: PROVIDER_NOT_FOUND
Description: This procedure returns a provider's Analysis Totals for a given month. Total records for months that have not yet been closed in Dentrix are identified with a '0' value returned for the Month and Year.
Example: call admin.sp_getprovidermonthtotals('DDS1',6,2011)

Column Name	Description	Type	Size
Month	Month	SMALLINT	2
Year	Year	SMALLINT	2
Provid	Provider ID	CHARACTER	4
beg_month	Beginning Month	SMALLINT	2
beg_year	Beginning Year	SMALLINT	2
total_charges	Total Charges	MONEY	10
charge_ins	Insurance Charges	MONEY	10
charges_adj	Charge Adjustments	MONEY	10
finance_charges	Finance Charges	MONEY	10
payments	Account Payments	MONEY	10
payments_ins	Insurance Payments	MONEY	10
payments_adj	Credit Adjustments	MONEY	10
chrgs_special_adj_mtd	Charge Special Adjustments	MONEY	10
pmts_special_adj_mtd	Credit Special Adjustments	MONEY	10
late_charges_mtd	MTD Late Charges	MONEY	10
late_charges_ytd	YTD Late Charges	MONEY	10
curr_pmt_agrmt_bal	Current Payment Agreement Balances	MONEY	10
rec0	Aged Balances 0-30 days	MONEY	10
rec30	Aged Balances 31-60 days	MONEY	10

rec60	Aged Balances 61-90 days	MONEY	10
rec90	Aged Balances over 90 days	MONEY	10
pay_plan	Future Due Payment Plan Balances	MONEY	10
current_credit_bal	Current Credit Balances	MONEY	10
beg_balance	Beginning Balance	MONEY	10
new_pat_count	New Patient Count	INTEGER	4
del_pat_count	Deleted Patient Count	INTEGER	4
ref_pat_count	Referred By Count	INTEGER	4
active_pat_count	Active Patient Count	INTEGER	4
total_pat_count	Total Patient Count	INTEGER	4
ins_pat_count	Insurance Patient Count	INTEGER	4
male_pat_count	Male Patient Count	INTEGER	4
fem_pat_count	Female Patient Count	INTEGER	4
under12_pat_count	Under 12 Patient Count	INTEGER	4
over65_pat_count	Over 65 Patient Count	INTEGER	4
family_count	Families	INTEGER	4
missed_pmt_count	Missed Payments	INTEGER	4
yr_beg_bal	Year Beginning Balance	MONEY	10
charges_ytd	YTD Charges	MONEY	10
charges_ins_ytd	YTD Insurance Charges	MONEY	10
charges_adj_ytd	YTD Adjustment Charges	MONEY	10
finance_chrg_ytd	YTD Finance Charges	MONEY	10
payments_ytd	YTD Account Payments	MONEY	10
payments_ins_ytd	YTD Insurance Payments	MONEY	10
payments_adj_ytd	YTD Credit Adjustments	MONEY	10
new_pat_count_ytd	YTD New Patient Count	INTEGER	4
del_pat_count_ytd	YTD Deleted Patient Count	INTEGER	4

ref_pat_count_ytd	YTD Referred By Count	INTEGER	4
missed_pmt_count_ytd	YTD Missed Payment Count	INTEGER	4
chrg_special_adj_ytd	YTD Charge Special Adjustments	MONEY	10
pmt_special_adj_ytd	YTD Credit Special Adjustments	MONEY	10

13.6 Provider Specialty

Stored Procedure: sp_getallproviderspecialties

Revision: 2

Inputs: None

Description: This procedure returns the provider specialties. A separate row will be returned for each provider specialty entered in the database.

Example: call admin.sp_getallproviderspecialties()

Column Name	Description	Type	Size
defid_id	Provider Specialty Definition ID	INTEGER	4
Descript	Provider Specialty Description	CHARACTER	52

14. Recall (Continuing Care)

14.1 Patient Recall

Stored Procedure Name: sp_getpatientrecalls
Revision: 3
Inputs: Patient GUID (CHARACTER)
Throws: PATIENT_NOT_FOUND
Description: This stored procedure returns a patient's recall information.
Example: call admin.sp_getpatientrecalls('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
modified_time_stamp	Modified Date/Time Stamp	TIMESTAMP	8
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
recall_type	Recall Type	CHARACTER	13
recall_description	Recall Description	CHARACTER	31
due_date	Recall Due Date	DATE	4
prior_date	Recall Prior Treatment Date	DATE	4
prov_id	Recall Provider ID	INTEGER	4
provider_last_name	Recall Provider Last Name	CHARACTER	21
provider_first_name	Recall Provider First Name	CHARACTER	16
recall_type_id	Recall Type ID	SMALLINT	2

14.2 All Patients' Recall Information

Stored Procedure Name: sp_getallpatientrecalls
Revision: 5
Inputs: None
Description: This procedure returns the recall information for all patients.
[CAUTION: This query could run slow depending on the number of patients]
Example: call admin.sp_getallpatientrecalls()

Column Name	Description	Type	Size

patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
recall_type	Recall Type	CHARACTER	13
recall_description	Recall Description	CHARACTER	31
due_date	Recall Due Date	DATE	4
prov_id	Recall Provider ID	INTEGER	4
provider_last_name	Recall Provider Last Name	CHARACTER	21
provider_first_name	Recall Provider First Name	CHARACTER	16
recall_type_id	Recall Type ID	SMALLINT	2

14.3 Recall Type

View Name: v_recall_type

Revision: 1

Inputs: None

Description: This procedure returns all recall types. A separate row will be returned for each recall type entered in the database.

Example: select * from admin.v_recall_type

Column Name	Description	Type	Size
automodifiedtimestamp	Modified Date/Time Stamp	TIMESTAMP	8
recallid	Recall ID	SMALLINT	2
patid	Patient ID	INTEGER	4
name	Recall Type Name	CHARACTER	13
descript	Recall Type Description	CHARACTER	31
flags	This is 0 unless this record is for a patient's override defaults for a continuing care type. Flags if the patients override defaults are for status, interval, providers and/or appointment time: - 16 (0x10)= interval is used to override - 32 (0x20)= provider is used to override - 48= interval and provider is used to override	SMALLINT	2

	- 64 (0x40)= appointment time is used to override - 80= interval and appointment time is used to override - 96= provider and appointment time is used to override - 112= interval, provider and appointment time is used to override - 128 (0x80)= status is used to override - 144= status and interval is used to override - 160= status and provider is used to override - 176= status, interval and provider is used to override - 192= status and appointment time is used to override - 208= status, interval and appointment time is used to override - 224= status, provider and appointment time is used to override - 240= status, interval, provider and appointment time is used to override		
color	Recall type color (HEX value)	BINARY	3
unit	Interval selection of days, weeks, months or years for this continuing care type: - 1= days - 2= weeks - 3= months - 4= years - 129= days with "+1 Day" checkbox is marked - 130= weeks with "+1 Day" checkbox is marked - 131= months with "+1 Day" checkbox is marked - 132= years with "+1 Day" checkbox is marked	TINYINT	1
qty	Number of days, weeks, months or years, according to Unit for this continuing care type. The maximum is 180.	TINYINT	1
provtype	Provider for this continuing care type: - 1= patient's Prov1 - 2= patient's Prov2 - 3= specific, selected provider	TINYINT	1

provid	Provider ID	CHARACTER	4
appttime	If this is not 0, indicates that a specific amount of time and time pattern has been set up for this continuing care type. The number of units.	SMALLINT	2
appttimepattern1	Bits to indicate the state of each time pattern checkbox for the number of units indicated for "ApptTime" (a checkbox for each time block). - Indicates chair time only if the checkbox is not marked - indicates only staff/assistant time if the checkbox is a slash (/) - indicates both provider and staff/assistant time if the checkbox is an X	TINYINT	1
appttimepattern2	See appttimepattern1	TINYINT	1
appttimepattern3	See appttimepattern1	TINYINT	1
appttimepattern4	See appttimepattern1	TINYINT	1
appttimepattern5	See appttimepattern1	TINYINT	1
appttimepattern6	See appttimepattern1	TINYINT	1
status	Continuing Care Status ID if a status is selected for the continuing care type (Links to the Practice Definitions table). May be 0.	SMALLINT	2

15. Referrals

15.1 Patient's Referrals

Stored Procedure Name: sp_getpatientreferrals

Revision: 3

Inputs: Patient GUID (CHARACTER)

Description: This procedure returns the Referred By information for a patient.

Example: call admin.sp_getpatientreferrals('045e8d6e-a74d-4bda-a9ec-0db9609ee61a')

Column Name	Description	Type	Size
referral_act_id	Referral Account ID	INTEGER	4
patient_id	Patient ID	INTEGER	4
patient_guid	Patient GUIID	CHARACTER	36
referral_id	Referral ID	INTEGER	4
last_name	Patient Last Name	CHARACTER	21
first_name	Patient First Name	CHARACTER	16
ref_last_name	Referral Last Name	CHARACTER	32
ref_first_name	Referral First Name	CHARACTER	16
ref_date	Referral Date	DATE	4
specialty_id	Referral Specialty ID	TINYINT	1
Title	Referral Title	CHARACTER	21
referral_type	Referral Type (Referred By, Referred To)	TINYINT	1
address_line1	Referral Address line 1	CHARACTER	31
address_line2	Referral Address line 2	CHARACTER	31
city	Referral City	CHARACTER	26
state	Referral State	CHARACTER	21
zipcode	Referral ZIP Code	CHARACTER	15
phone	Referral Phone number	CHARACTER	17
ext	Referral	CHARACTER	5
other_phone_number	Referral other phone number	CHARACTER	17

fax_number	Referral fax number	CHARACTER	17
email_address	Referral e-mail address	CHARACTER	60

15.2 Referral Source

Stored Procedure Name: sp_getreferralsource
Revision: 2
Inputs: None
Description: This procedure returns information for all referral sources.
Example: call admin.sp_getreferralsource()

Column Name	Description	Type	Size
ref_type	Referral Type: - 0=Referred by patient - 1=Referred by doctor/other source - 2=Referred to	TINYINT	1
ref_id	Referral ID	INTEGER	4
ref_last_name	Referral Last Name	CHARACTER	32
ref_first_name	Referral First Name	CHARACTER	16
ref_mi	Referral Middle Initial	CHARACTER	2
ref_specialty	Referral Specialty	CHARACTER	52
salutation	Salutation for correspondence	CHARACTER	32
Title	Title (Mr., Mrs., etc.)	CHARACTER	21
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
city	City	CHARACTER	26
state	State	CHARACTER	21
zipcode	ZIP Code	CHARACTER	15
phone	Phone	CHARACTER	17
other_phone	Other Phone	CHARACTER	17
fax	Fax	CHARACTER	17
email_address1	Primary Email Address	CHARACTER	60

email_address2	Secondary Email Address	CHARACTER	60
non_person_flag	Non-Person Flag: - 0=Person - 1=Non-Person	BIT	1
contact_id	Contact ID. Username from DXWeb referral setup.	CHARACTER	20
send_email	Flag for "Send E-mail Notification": - 0=Do not Send E-mail Notification - 1=Send E-mail Notification	BIT	1
use_email	Flag to indicate to use primary e-mail address or secondary. - 0=Use Primary E-mail address - 1=Use Secondary E-mail address	BIT	1
notes	Referral Notes	VARCHAR	999

15.3 Referral Specialty

View Name: v_referral_specialty

Revision: 1

Inputs: None

Description: This procedure returns the referral specialties. A separate row will be returned for each referral specialty entered in the database.

Example: select * from admin.v_referral_specialty

Column Name	Description	Type	Size
defid_id	Referral Specialty Definition ID	INTEGER	4
description	Referral Specialty Description	CHARACTER	52

16. Staff Information

16.1 Staff

View Name: v_staff
Revision: 1
Description: This view retrieves staff information.
Example: select * from admin.v_staff

Column Name	Description	Type	Size
staff_id	Staff ID	CHARACTER	4
last_name	Staff last name	CHARACTER	21
first_name	Staff first name	CHARACTER	16
mi	Staff middle initial	CHARACTER	2
title	Staff title	CHARACTER	21
work_phone	Staff work phone number	CHARACTER	17
work_phone_ext	Staff work phone extension	CHARACTER	5
address_line1	Address line 1	CHARACTER	31
address_line2	Address line 2	CHARACTER	31
city	City	CHARACTER	26
state	State	CHARACTER	21
zip_code	ZIP Code	CHARACTER	15
ssn	Social Security Number	CHARACTER	10
isnonperson	Non-person Flag	BIT	1
inactive	Inactive Flag (0=Active; 1=Inactive)	BIT	1

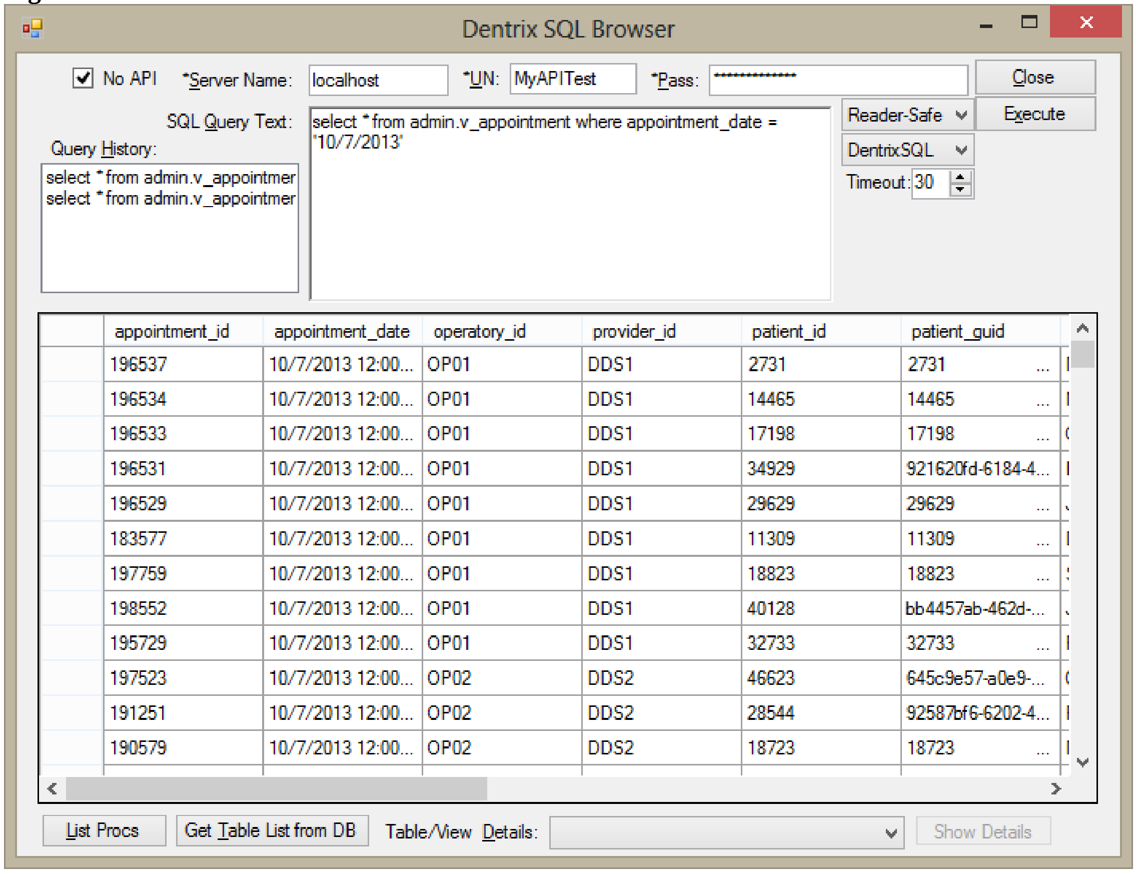
Using the Dentrix SQL Browser Tool

The Dentrix SQL Browser tool can be used to quickly test your connection to the API (using your API username and password) as well as build and make SQL queries to a Dentrix G5 and above database. The Dentrix SQL Browser requires that Dentrix G5 be installed on computer from which it is run in order to connect to the database. The Dentrix SQL Browser can be downloaded from the Developer Program website (see Online Resources section of this document).

Enter your API username in the "UN" field and password in the "Pass" field. Then, enter your query in the "SQL Query Test" field and click the Execute button. Or, if you simple want to see the tables and columns that are available in the Views, Stored Procedures or "raw" tables, just click the "Get Table List from DB" button at the bottom of the window (after you have entered your username and password). Then, for a selected Table, View or raw database table, click "Show Details".

1. To launch the Dentrix SQL Browser, locate and run DtxSQLBrowser.exe.
2. Once the Dentrix SQL Browser opens (see **Figure 3**), enter the API username and password that you received upon registering for the Dentrix Developer Programs in the 'UN' and 'Pass' fields. *Note: The username and password used for the last session (if applicable) will be populated in the UN and Pass fields by default.*

Figure 3



No API: ✓ *Server Name: localhost *UN: MyAPITest *Pass: **************

SQL Query Text: select * from admin.v_appointment where appointment_date = '10/7/2013'

Reader-Safe · DentrixSQL · Timeout: 30

Query History:
select * from admin.v_appointmer
select * from admin.v_appointmer

appointment_id	appointment_date	operatory_id	provider_id	patient_id	patient_guid
196537	10/7/2013 12:00...	OP01	DDS1	2731	2731 ...
196534	10/7/2013 12:00...	OP01	DDS1	14465	14465 ...
196533	10/7/2013 12:00...	OP01	DDS1	17198	17198 ...
196531	10/7/2013 12:00...	OP01	DDS1	34929	921620fd-6184-4...
196529	10/7/2013 12:00...	OP01	DDS1	29629	29629 ...
183577	10/7/2013 12:00...	OP01	DDS1	11309	11309 ...
197759	10/7/2013 12:00...	OP01	DDS1	18823	18823 ...
198552	10/7/2013 12:00...	OP01	DDS1	40128	bb4457ab-462d-...
195729	10/7/2013 12:00...	OP01	DDS1	32733	32733 ...
197523	10/7/2013 12:00...	OP02	DDS2	46623	645c9e57-a0e9-...
191251	10/7/2013 12:00...	OP02	DDS2	28544	92587bf6-6202-4...
190579	10/7/2013 12:00...	OP02	DDS2	18723	18723 ...

3. By default, the 'No API' checkbox is checked and the 'Server Name' field is populated with *localhost*. This is the correct setting if you are running the Dentrix SQL Browser from the Dentrix server. If you are running the Dentrix SQL Browser from a machine that is not the Dentrix server, you will need to uncheck the 'No API' checkbox (which will cause the 'Server Name' field to be hidden) so that the program will automatically detect Dentrix server machine location.
4. Enter your SQL statement in the 'SQL Query Text' field or double-click a previously saved SQL statement from the 'Query History' list.
5. Choose one of the following options from the 'Reader-Safe' drop-down to determine how the data will be read and displayed in the data grid:
    - **Reader** – using the ODBC driver, reads the results for each row (one a time) sequentially and fills the data grid accordingly
    - **Adapter** – using the ODBC driver, reads the entire result set and then fills the data grid
    - **Reader-Safe** - using the ODBC driver, reads the results for each row (one a time) sequentially (with NULL checking) and fills the data grid accordingly
    - **Faircom Reader** – using the Faircom® ODBC driver, reads the results for each row (one a time) sequentially and fills the data grid accordingly
    - **Faircom Adapter** - using the Faircom® ODBC driver, reads the entire result set and then fills the data grid
    - **Faircom Reader-Safe** - using the Faircom® ODBC driver, reads the results for each row (one a time) sequentially (with NULL checking) and fills the data grid accordingly
6. Choose one of the following options from the 'DentrixSQL' drop-down to determine the database you wish to query:
    - **DentrixSQL** – queries the default "live" Dentrix database on the local machine or Dentrix server machine on the network according to the specified 'Server Name'.
    - **TutorSQL** – queries the "sample" Dentrix database on the local machine or Dentrix server machine on the network according to the specified 'Server Name'.
7. Change the 'Timeout' setting, as needed, to specify the timeout value (in seconds) for the SQL command execution.
8. To view all available stored procedures in the API that you have access to, click the 'List Procs' button and then click the 'All Stored Procedures' drop-down below the data grid.
9. To view all available tables / table views in the API that you have access to, click the 'Get Table List from DB' button and then click the 'Table/View Details' drop-down below the data grid. To view the schema details for a given table / table view, select it from the 'Table/View Details' drop-down and then click the 'Show Details' button. The schema for the select table/view will be displayed in the data grid.
10. Click 'Execute' (or hit Enter) to execute the query specified in the 'SQL Query Text' box.

Using the Dentrix SQL Browser Source Code

The Dentrix SQL Browser source code is provided to you to assist you in your development efforts and can be found in the 'Dentrix SQL Browser Source' folder in the *DentrixSDK_2.1.zip* file accessible from the Developer Program website (see Online Resources section of this document for details). We recommend that you review the following two functions defined in the source code below and consider incorporating these methods in your application's code:

1. **Retrieving ODBC connection string at run-time**

   This sample code demonstrates how to get the ODBC connection string for your application at run-time.

2. **Data Retrieving methods**

   This sample code demonstrates using the following 'data retrieving' methods to read the data result set returned from your executed SQL queries: Reader, Adapter, Reader-Safe, Faircom Reader, Faircom Adapter and Faircom Reader-Safe. ***NOTE: It is recommended that you use the Faircom Reader-Safe method to ensure that any NULL references that may exist in your Dentrix customers' databases are handled properly in your code.***

The following section of code is taken from MainForm.cs found in the DentrixSQLBrowswerSource.zip accessible from ddp.dentrix.com.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Odbc;
using System.Data.Common;
using System.Runtime.InteropServices;
using Ctree.Data.SqlClient;

namespace DtxSQLBrowser
{
    public partial class MainForm : Form
    {

        public enum DtxReaderType
        {
            Reader = 0,
            Adapter = 1,
            ReaderSafe = 2,
            FaircomReader = 3,
            FaircomAdapter = 4,
            FaircomReaderSafe = 5
        }

        #region  Properties
        static DbConnection _connection;
        static string _connectionStringBase =
"host={0};UID={1};PWD={2};Database={3};DSN=c-treeACE ODBC Driver;port=6597";
        static string _connectionStringBaseCtree =
"host={0};UID={1};PWD={2};Database={3};port=6597";
        public string DB
        {
            get
            {
                return dataOrTutorDrop.SelectedItem.ToString();
            }
        }
        public string UserName
        {
            get
            {
                return userNameTB.Text;
            }
        }
        public string Password
        {
            get
            {
                return passwordTB.Text;
            }
        }
        public string Host
        {
            get
            {
                return HostTB.Text;
            }
        }
        public string QueryText
        {
            get
            {
                return queryTextTB.Text;
            }
        }
        public int Timeout
        {
            get
            {
                return (int)timeoutNumBox.Value;
            }
        }
        public string Cur_User
        {
            get;
            set;
        }
        public string Cur_Pass
        {
            get;
            set;
        }
        public string Cur_Host
        {
            get;
            set;
        }
        public bool MissingInfo
        {
            get
            {
                if (UserName.TrimEnd() == String.Empty || Password.TrimEnd() ==
String.Empty || Host.Trim() == String.Empty)
                    return true;
                return false;
            }
        }
        public DtxReaderType ConnectionType
        {
            get
            {
                return (DtxReaderType)connectionTypeDropB.SelectedIndex;
            }
        }
        public bool NeedsOdbcConnection
        {
            get
            {
                switch (ConnectionType)
                {
                    case DtxReaderType.Adapter:
                    case DtxReaderType.Reader:
                    case DtxReaderType.ReaderSafe:
                        return true;
                    default:
                        return false;
                }
            }
        }
        public bool isCtreeConnection
        {
            get { return (_connection != null && _connection is CtreeSqlConnection); }
        }
        public bool isOdbcConnection
        {
            get { return (_connection != null && _connection is OdbcConnection); }
        }
        #endregion

        #region ctor
        public MainForm()
        {
            InitializeComponent();
            this.StartPosition = FormStartPosition.WindowsDefaultBounds;

            if (Properties.Settings.Default.WindowPosition != Rectangle.Empty)
            {
                this.StartPosition = FormStartPosition.Manual;
                this.DesktopBounds = Properties.Settings.Default.WindowPosition;

                this.WindowState = Properties.Settings.Default.WindowState;
            }
            else
            {
                this.StartPosition = FormStartPosition.CenterScreen;
            }

            HostTB.Text = Properties.Settings.Default.lastServerName;
            userNameTB.Text = Properties.Settings.Default.lastUserName;
            passwordTB.Text = Properties.Settings.Default.lastUserPassword;
            if (Properties.Settings.Default.QueryHistory != null)

queryHistory.Items.AddRange(Properties.Settings.Default.QueryHistory.Cast<string>().ToArr
ay());
        }
        #endregion

        #region Business Logic
        /// <summary>
        /// Creates a connection to the Dentrix Database using the Credentials provided
by the UI
        /// </summary>
        private DbConnection GetDentrixConnection()
        {
            if (_connection == null || (isCtreeConnection && NeedsOdbcConnection) ||
(isOdbcConnection && !NeedsOdbcConnection))
            {
                _connection = GetConnectionObject();
            }
            if (CredentialsChanged())
            {
                if (_connection.State == ConnectionState.Open || _connection.State ==
ConnectionState.Broken)
                    _connection.Close();

                Properties.Settings.Default.lastUserName = Cur_User = UserName;
                Properties.Settings.Default.lastUserPassword = Cur_Pass = Password;
                Properties.Settings.Default.lastServerName = Cur_Host = Host;
                Properties.Settings.Default.Save(); //updates the 'app settings object'
so that next load will remember these settings
            }
            if (_connection.State == ConnectionState.Closed)
            {

                if (noAPI_CB.Checked)
                {
                    if (ConnectionType == DtxReaderType.FaircomAdapter || ConnectionType
== DtxReaderType.FaircomReader || ConnectionType == DtxReaderType.FaircomReaderSafe)
                    {
                        _connection.ConnectionString =
string.Format(_connectionStringBaseCtree, Cur_Host, Cur_User, Cur_Pass, DB);
                    }
                    else
                    {
```

```
            _connection.ConnectionString = 
string.Format(_connectionStringBase, Cur_Host, Cur_User, Cur_Pass, DB);
          }
        }
1       else
        {
          //Should call into the DentrixAPI.dll
          StringBuilder connectionString = new StringBuilder(512);
          lock (connectionString)
          {
            switch (ConnectionType)
            {
              case DtxReaderType.Adapter:
              case DtxReaderType.Reader:
              case DtxReaderType.ReaderSafe:
                DENTRIXAPI_GetConnectionString(Cur_User, Cur_Pass, 
connectionString, 512);
                _connection.ConnectionString = 
connectionString.ToString();
                break;
              case DtxReaderType.FaircomAdapter:
              case DtxReaderType.FaircomReader:
              case DtxReaderType.FaircomReaderSafe:
                DENTRIXAPI_GetFCConnectionString(Cur_User, Cur_Pass, 
connectionString, 512);
                _connection.ConnectionString = 
connectionString.ToString();
                break;

            }
          }
        }
        _connection.Open();
      }
      return _connection;
    }

    private DbConnection GetConnectionObject()
    {

      if (ConnectionType == DtxReaderType.FaircomAdapter || ConnectionType == 
DtxReaderType.FaircomReader || ConnectionType == DtxReaderType.FaircomReaderSafe)
      {
        return new CtreeSqlConnection();
      }
      else
      {
        return new OdbcConnection();
      }
    }
    /// <summary>
    /// Method to ensure connection details have changed or not
    /// </summary>
    private bool CredentialsChanged()
    {
      if (Cur_User != UserName || Cur_Pass != Password || Cur_Host != Host)
        return true;
      else
```

```
                return false;
        }
        /// <summary>
        /// Message to inform user that required data fields were not provided
        /// </summary>
        private void ShowInvalidDataMessage()
        {
            MessageBox.Show("One or more required fields were not provided.");
        }
        /// <summary>
        /// Creates command with the UI settings provided and returns it.
        /// </summary>
        private DbCommand ConstructCommand()
        {
            DbCommand cmd = _connection.CreateCommand();
            cmd.CommandText = QueryText;
            cmd.CommandTimeout = Timeout;
            return cmd;
        }
        /// <summary>
        /// Execute Query according to the settings/values provided in UI.
        /// It is expected that _connection is already established and open.
        /// </summary>
        private void ExecuteQuery()
        {
            DataTable dtable = new DataTable();
            switch (ConnectionType)
```

2

```
            {
               case DtxReaderType.ReaderSafe:
                case DtxReaderType.FaircomReaderSafe:
                    {
                        DbCommand command = ConstructCommand();
                        DbDataReader reader = command.ExecuteReader();
                        bool columnsBuilt = false;
                        while (reader.Read())
                        {
                            if (columnsBuilt == false)
                            {
                                BuildColumnData(dtable, reader);
                                columnsBuilt = true;
                            }
                            AddDataRow(dtable, reader);
                        }
                        reader.Close();
                        break;
                    }
                case DtxReaderType.Reader:
                case DtxReaderType.FaircomReader:
                    {
                        DbCommand command = ConstructCommand();
                        DbDataReader reader = command.ExecuteReader();
                        dtable.Load(reader, LoadOption.OverwriteChanges);
                        reader.Close();
                        break;
                    }
                case DtxReaderType.FaircomAdapter:
                    {
                        using (Ctree.Data.SqlClient.CtreeSqlDataAdapter adap = new
CtreeSqlDataAdapter(QueryText, (CtreeSqlConnection)_connection))
                        {
                            dtable.BeginInit();
                            dtable.BeginLoadData();
                            adap.Fill(dtable);
                            dtable.EndLoadData();
                            dtable.EndInit();
                            adap.Dispose();
                            _connection.Close(); //Need to override the close on this so
that you can execute a reader following this guy.
                        }
                        break;
                    }
                default:
                case DtxReaderType.Adapter:
                    {
                        using (OdbcDataAdapter adap = new OdbcDataAdapter(QueryText,
(OdbcConnection)_connection))
                        {
                            dtable.BeginInit();
                            dtable.BeginLoadData();
                            adap.Fill(dtable);
                            dtable.EndLoadData();
                            dtable.EndInit();
                            adap.Dispose();
                        }
                        break;
                    }

              //}  end switch
            }
            dataGrid.DataSource = dtable;
        }

        /// <summary>
        /// Builds DataColumns according to reader
        /// </summary>
        /// <param name="table"></param>
        /// <param name="reader"></param>
        private void BuildColumnData(DataTable table, DbDataReader reader)
        {
            for (int i = 0; i < reader.FieldCount; i++)
            {
                DataColumn dc = new DataColumn();
                dc.ColumnName = reader.GetName(i);
                table.Columns.Add(dc);
            }
        }

        /// <summary>
        /// Add Data Row via Reader to Table
        /// </summary>
        /// <param name="table">Table in which the row should be added</param>
        /// <param name="reader">reader object with the particular row loaded and ready
for read</param>
        private void AddDataRow(DataTable table, DbDataReader reader)
        {
            DataRow row = table.NewRow();
            for (int i = 0; i < reader.FieldCount; i++)
            {
                if ( reader[i] != DBNull.Value)
                    row[i] = reader[i];
            }
            table.Rows.Add(row);
        }

        /// <summary>
        /// Request Stored Procedure List and filter by USER to execute rights
        /// </summary>
        private void ReadValidStoredProcedures()
        {
            //Clear existing Table/SProcs list
            tableViewLB.Items.Clear();

            DbCommand command = _connection.CreateCommand();
            command.CommandText = string.Format("select * from admin.systabauth where 
grantee='{0}' and exe='y' Order By Tbl", UserName);
            DbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                tableViewLB.Items.Add(reader[3].ToString());
            }
            reader.Close();
        }

        /// <summary>
        /// Request Table List and filter by USER to read rights
        /// </summary>
        private void ReadTablesAndViews()
        {
            //Clear existing Table/SProcs list
            tableViewLB.Items.Clear();

            DbCommand command = _connection.CreateCommand();
            command.CommandText = string.Format("select * from admin.systabauth where 
grantee='{0}' and sel='y' ORDER BY tbl", UserName);

            DbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                tableViewLB.Items.Add(reader[3].ToString());
            }
            reader.Close();
        }

        /// <summary>
        /// Get Table Schema for selected table and show all details in datagrid
        /// </summary>
        private void GetTableSchema()
        {
            if (tableViewLB.Text.TrimEnd() == "")
                return;

            DbCommand command = _connection.CreateCommand();
            command.CommandText = string.Format("select TOP 1 * from admin.{0}",
tableViewLB.Text);
            command.CommandTimeout = (int)timeoutNumBox.Value;
            DbDataReader reader = command.ExecuteReader();
            DataTable table = reader.GetSchemaTable();
            dataGrid.DataSource = table;
        }
        #endregion

        #region Event Handlers
        /// <summary>
        /// Here to select default selections
        /// </summary>
        private void MainForm_Load(object sender, EventArgs e)
        {
            dataOrTutorDrop.SelectedIndex = 0; //select 'Data'
            connectionTypeDropB.SelectedIndex = 2; // ODBC Adapter
        }
        /// <summary>
        /// Execute query button has been clicked.  Validate details then run query.
        /// </summary>
        private void execute_Click(object sender, EventArgs e)
        {
            if (MissingInfo == true)
            {
                ShowInvalidDataMessage();
                return;
            }

            try
            {
                GetDentrixConnection();
                using (_connection)
                {
                    ExecuteQuery();
                }
                //Save query to history
                if (string.IsNullOrWhiteSpace(queryTextTB.Text) == false)
                {
                    if (queryHistory.Items.Contains(queryTextTB.Text) == false)
                    {
                        queryHistory.Items.Insert(0, queryTextTB.Text); //always insert
as first
                    }
                    else //Item is already in list.  Move to first
                    {

                        queryHistory.Items.Remove(queryTextTB.Text);
                        queryHistory.Items.Insert(0, queryTextTB.Text);
                    }
                    //update user settings object with query history and save it
                    SaveHistoryCollection();
                }
            }
            catch (Exception ex)
            {
                _connection.Close(); //In the event the query failed, just close the
connection so that a new
```

```
                //connection is started at the start of the next query.
                MessageBox.Show("Error: " + ex.Message);
            }
        }

        private void SaveHistoryCollection()
        {
            Properties.Settings.Default.QueryHistory = new
System.Collections.Specialized.StringCollection();

Properties.Settings.Default.QueryHistory.AddRange(queryHistory.Items.Cast<string>().ToArr
ay());
            Properties.Settings.Default.Save();
        }
        /// <summary>
        /// Restores the 'saved' query to the query text box
        /// </summary>
        private void queryHistory_DoubleClick(object sender, EventArgs e)
        {
            //pass the selected query to the query tb (if any)
            if (queryHistory.SelectedIndex != -1)
            {
                queryTextTB.Text = (string)queryHistory.SelectedItem;
            }
        }
        /// <summary>
        /// If entered via Mneumonic, we need need to catch 'enter' clicks and send the
current query to the tb
        /// </summary>
        private void queryHistory_KeyUp(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter || e.KeyCode == Keys.Return)
            {
                queryHistory_DoubleClick(null, null); //Just fire existing event for
triggering this behavior.
            }
            //Also allow deletion of items in the list.
            if (e.KeyCode == Keys.Delete)
            {
                queryHistory.Items.RemoveAt(queryHistory.SelectedIndex);
                SaveHistoryCollection();
            }
        }
        /// <summary>
        /// When the history item index changes, update the tooltip text with the text
you want
        /// </summary>
        private void queryHistory_SelectedIndexChanged(object sender, EventArgs e)
        {
            toolTip.SetToolTip(queryHistory, (string)queryHistory.SelectedItem);
        }
        /// <summary>
        /// Get Procedures List btn has been clicked. Validate connection data then
fetch.
        /// </summary>
        private void fetchProceduresBtn_Click(object sender, EventArgs e)
        {
            if (MissingInfo == true)
            {
                ShowInvalidDataMessage();
                return;
            }
            try
            {
                GetDentrixConnection();
                using (_connection)
                {
                    ReadValidStoredProcedures();
                    if (tableViewLB.Items.Count > 0)
                        tableViewLB.SelectedIndex = 0;
                    fetchProceduresBtn.Enabled = false;
                    getTableListBtn.Enabled = true;
                    detailLabel.Text = "All Stored Procedures:";
                }
                showDetailsBtn.Enabled = false;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
        /// <summary>
        /// Get Table List btn has been clicked. Validate connection data then fetch.
        /// </summary>
        private void getTableListBtn_Click(object sender, EventArgs e)
        {
            if (MissingInfo == true)
            {
                ShowInvalidDataMessage();
                return;
            }
            try
            {
                GetDentrixConnection();
                using (_connection)
                {
                    ReadTablesAndViews();
                    if (tableViewLB.Items.Count > 0)
                        tableViewLB.SelectedIndex = 0;
                    fetchProceduresBtn.Enabled = true;
                    getTableListBtn.Enabled = false;
                    detailLabel.Text = "Table/View Details:";
                }
                showDetailsBtn.Enabled = true;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
        /// <summary>
        /// Show Detail Btn has been clicked. Validate connection data then fetch.
        /// </summary>
        private void showDetailsBtn_Click(object sender, EventArgs e)
        {
            if (MissingInfo == true)
            {
                ShowInvalidDataMessage();
                return;
            }
            try
            {
                GetDentrixConnection();
                using (_connection)
                {
                    GetTableSchema();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
        }
        /// <summary>
        /// Close Btn has been clicked.  Close current connection, if any, then close
Dialog.
        /// </summary>
        private void close_Click(object sender, EventArgs e)
        {
            if (_connection != null && _connection.State == ConnectionState.Open)
                _connection.Close();
            Close();
        }
        /// <summary>
        /// In order to allow response time from the server there must be a min timeout
of 15.  It is set this
        /// high because large querries can take up to this amount of time to return.
        /// </summary>
        private void timeoutNumBox_Leave(object sender, EventArgs e)
        {
            if (Timeout < 15)
            {
                MessageBox.Show("Value must be at least 15 seconds to ensure adequate
time for results to return.");
                timeoutNumBox.Value = 15;
            }
        }
        /// <summary>
        /// Select All text in Query Box - Needed for easier Automation
        /// </summary>
        private void queryTextTB_Enter(object sender, EventArgs e)
        {
            queryTextTB.SelectAll();
            //set the default button to execute
            this.AcceptButton = execute;
        }
        /// <summary>
        /// Clears the Accept button focus so that ENTER no longer triggers a query
        /// </summary>
        private void queryTextTB_Leave(object sender, EventArgs e)
        {
            this.AcceptButton = null; //Clear the default button click
        }
        private void queryTextTB_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Alt && e.KeyCode == Keys.Enter)
            {
                queryTextTB.Text += "\n";
                queryTextTB.Select(queryTextTB.Text.Length - 1, 0);
            }
        }
        /// <summary>
        /// Enables/Disables the usage of the 'HOST' text box when 'using api' is
selected.
        /// </summary>
        private void useManualCB_CheckedChanged(object sender, EventArgs e)
        {
            if (noAPI_CB.Checked)
            {
                HostTB.Visible = true;
                dataOrTutorDrop.Enabled = true;
            }
            else
            {
                HostTB.Visible = false;
                dataOrTutorDrop.Enabled = false;
                dataOrTutorDrop.SelectedIndex = 0;
            }
        }
        /// <summary>
        /// Save position state info for next session
        /// </summary>
        private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
        {
            if (this.WindowState != FormWindowState.Minimized)
            {
                Properties.Settings.Default.WindowState = this.WindowState;
            }
            Properties.Settings.Default.WindowPosition = this.DesktopBounds;

            Properties.Settings.Default.Save();
        }
        #endregion

        #region DLL Import Methods
        [DllImport("Dentrix.API.dll", CharSet = CharSet.Ansi, CallingConvention =
CallingConvention.StdCall)]
        static extern void
DENTRIXAPI_GetConnectionString([MarshalAs(UnmanagedType.LPStr)]string szUserId,
[MarshalAs(UnmanagedType.LPStr)]string szPassword, StringBuilder szConnectionsString, int
ConnectionStringSize);

        [DllImport("Dentrix.API.dll", CharSet = CharSet.Ansi, CallingConvention =
CallingConvention.StdCall)]
        static extern void
DENTRIXAPI_GetFCConnectionString([MarshalAs(UnmanagedType.LPStr)]string szUserId,
[MarshalAs(UnmanagedType.LPStr)]string szPassword, StringBuilder szConnectionsString, int
ConnectionStringSize);

        #endregion
    }
}
```

Table View Tutorial

The following Table View tutorial demonstrates how to develop a simple application step-by-step for an available Table View using C#. The project files for this tutorial are also found in the 'Tutorials\ViewTutorial' folder in the *DentrixSDK_2.1.zip* file accessible from the Developer Program website (see Online Resources section of this document for details).

To successfully use and compile the Table View tutorial, the following prerequisites must be satisfied:

- The following must be installed:
  - Dentrix G5 or higher
  - Microsoft® Visual Studio 2008 or higher
  - Microsoft® .NET 3.5 or higher
- One of the methods for requesting access to the Dentrix API must be completed (after installed Dentrix G5 or higher)
- Knowledge of the C# programming language is recommended

We will use the Practice Prescriptions table view (v_rxlist) in the tutorial below.

Create the Table View Tutorial project

1. Using Microsoft® Visual Studio 2008, create a new project using the Console Application template and name it "TutorialView".

   This will create a solution called TutorialView and a project with the same name. This project includes a file called Program.cs.
2. In program.cs, make sure the project includes references to the following three namespaces:

```
using System;
using System.Data.Odbc;
```

3. Add the following members to the class TutorialView:

```
// ODBC declare connection, command and reader objects
    static OdbcConnection connection;
    static OdbcCommand command;
    static OdbcDataReader reader;
```

4. Modify the Main() function to include calls to the following methods:

```
// ODBC declare connection, command and reader objects
    static OdbcConnection connection;
    static OdbcCommand command;
    static OdbcDataReader reader;

static void Main()
{
        Initialize();
        CallView();
        DisplayRecords();
        connection.Close();
        Console.Write("\nPress <ENTER> key to exit . . .");
        Console.ReadLine();
}
```

5. Create a method called Initialize and add the calls to open the database connection.

```
    /// <summary>
    /// Initialize the database
    /// </summary>
    static void Initialize()
    {
        try
             {
                     // initialize ODBCConnection object
                     connection = new OdbcConnection();



                     string currentDB = GetCurrentDB();


                   // build the string using the arguments passed when calling this
        exe
             connection.ConnectionString =
    "UID=<username>;PWD=<password>;Database="+currentDB+";host=<serverName>;DSN=c-
    treeACE ODBC Driver;port=6597";

    // initialize command object
    command = new OdbcCommand();
    command.CommandType = System.Data.CommandType.Text;
    command.Connection = connection;
    // connect to server
    connection.Open();
    }
    catch (Exception e)
        {
        Console.WriteLine("Error: " + e.Message);
        Environment.Exit(1);
    }
}

/// <summary>
/// This method returns the database that is currently used.
/// </summary>
/// <returns></returns>
static string GetCurrentDB()
{
    /// This is the default Dentrix database
    return "DentrixSQL";
}


    try
    {
        // create table
        Console.WriteLine("\tCalling v_rxlist...");
        command.CommandText = "select * from admin.v_rxlist";
        command.ExecuteNonQuery();
    }
    catch (Exception e)
```

7. Create a method to call the view and call it “CallView”.

```
/// <summary>
/// CallView:  This function executes the select command
/// </summary>
static void CallView()
{
```

8. Create a function to display records and call it “DisplayRecords”.

```
    {
        Console.WriteLine("Error: " + e.Message);
        Environment.Exit(1);
    }
}

/// <summary>
/// DisplayRecords: This function displays the records in the view if any.
/// </summary>
static void DisplayRecords()
{
    try
    {
        reader = (OdbcDataReader)command.ExecuteReader();
        // read the returned resultset
        while (reader.Read())
        {
            /*
            * This view consists the following fields:
            * rxdef_id (TINYINT)
            * drug_name (CHARACTER 50)
            * description (CHARACTER 50)
            * rx_date (DATE)
            * For the purpose of keeping it simple, we are displaying the rxdef_id
            and drug_name on the console (the first two fields)
            */
            Console.WriteLine("\n\t\t{0}  {1}  ", reader.GetInt16(0),
            reader.GetString(1));
            }
            // close the reader
            reader.Close();
        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e.Message);
            Environment.Exit(1);
        }
    }
}
```

9. Compile and run the program. You should see a Command Prompt window open with the prescription information.

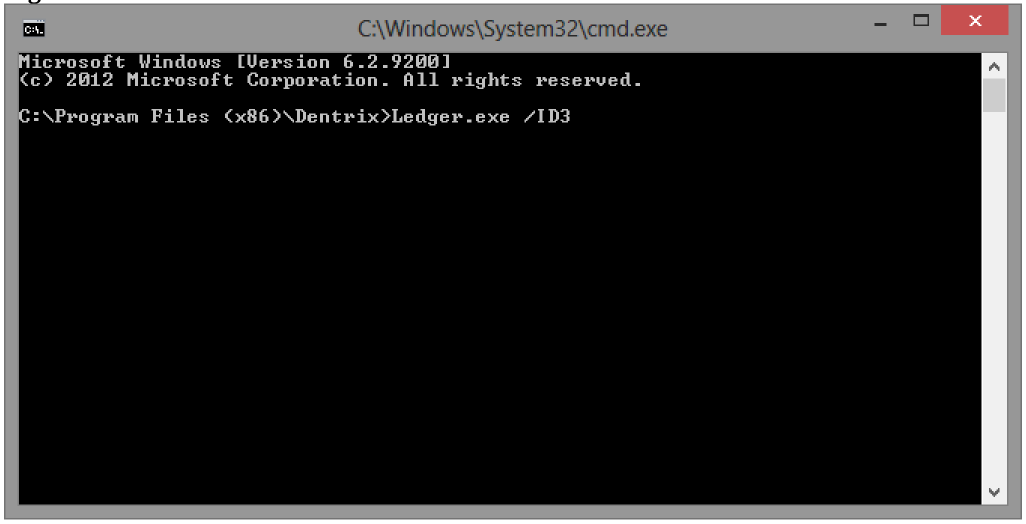
Opening Dentrix with Patient Selected

As part of your app's integration with Dentrix, you may wish to launch a specific Dentrix module with a specified patient selected. This can be accomplished by opening the module via a command line formatted as shown below:

```
<Drive letter>\<Dentrix program files path>\<Dentrix module executable name>.exe /ID<Patient ID>
```

Below is an example of how this is done manually from a command prompt (**Figure 4**). In this example, the Ledger will be launched for a patient named *Joe Patient*. Joe's patient ID in the Dentrix database is **3**. Using this method, the Ledger is opened with Joe's information as illustrated in **Figure 5**.

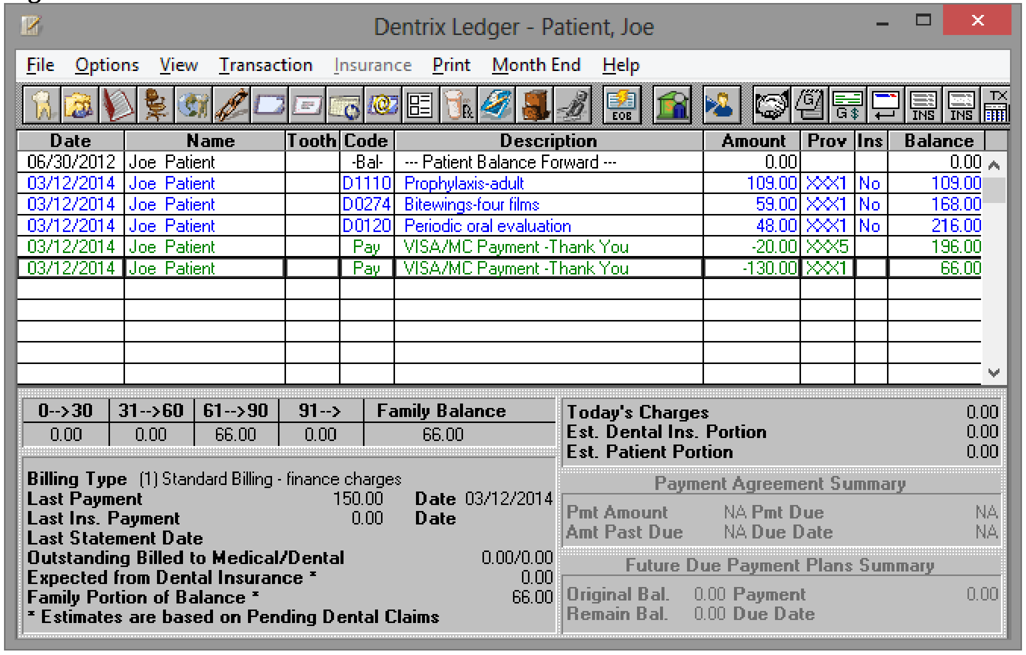
Figure 4



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Dentrix>Ledger.exe /ID3
```

Figure 5


Dentrix Ledger - Patient, Joe
File Options View Transaction Insurance Print Month End Help
EOB
INS
INS
TX

Date	Name	Tooth	Code	Description	Amount	Prov	Ins	Balance
06/30/2012	Joe Patient		-Bal-	--- Patient Balance Forward ---	0.00			0.00
03/12/2014	Joe Patient		D1110	Prophylaxis-adult	109.00	XXX1	No	109.00
03/12/2014	Joe Patient		D0274	Bitewings-four films	59.00	XXX1	No	168.00
03/12/2014	Joe Patient		D0120	Periodic oral evaluation	48.00	XXX1	No	216.00
03/12/2014	Joe Patient		Pay	VISA/MC Payment -Thank You	-20.00	XXX5		196.00
03/12/2014	Joe Patient		Pay	VISA/MC Payment -Thank You	-130.00	XXX1		66.00

0-->30	31-->60	61-->90	91-->	Family Balance
0.00	0.00	66.00	0.00	66.00

Today's Charges	0.00
Est. Dental Ins. Portion	0.00
Est. Patient Portion	0.00

Billing Type (1) Standard Billing - finance charges
Last Payment 150.00 **Date** 03/12/2014
Last Ins. Payment 0.00 **Date**
Last Statement Date
Outstanding Billed to Medical/Dental 0.00/0.00
Expected from Dental Insurance * 0.00
Family Portion of Balance * 66.00
*** Estimates are based on Pending Dental Claims**

Payment Agreement Summary
Pmt Amount NA Pmt Due NA
Amt Past Due NA Due Date NA

Future Due Payment Plans Summary
Original Bal. 0.00 Payment 0.00
Remain Bal. 0.00 Due Date

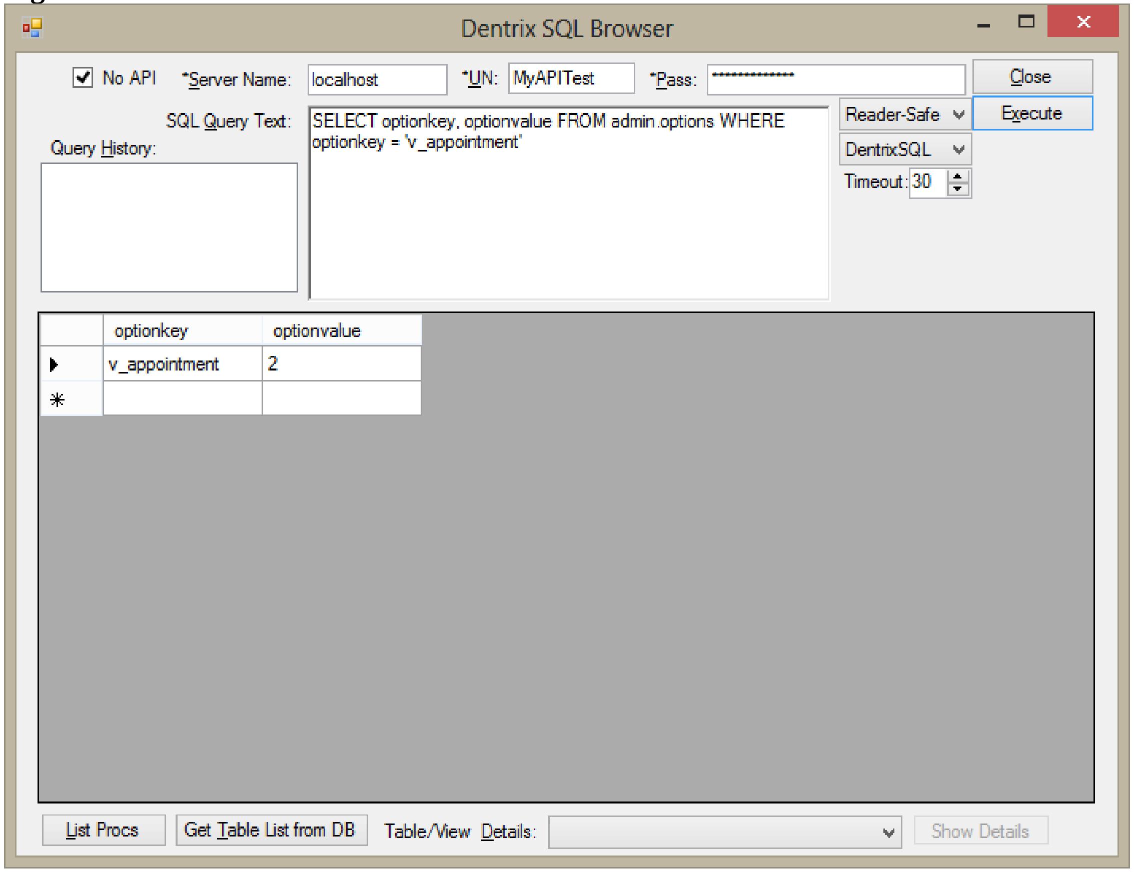
Checking Version of Installed API

As of Dentrix G5.2, the version (revision) of each Table View and Stored Procedure in the installed API are stored in *admin.options* table in the Dentrix database. To check the version of Table View or Stored Procedure, you should execute a query like the following, where the version (revision) number is returned in the **optionvalue** column:

```
SELECT optionkey, optionvalue FROM admin.options WHERE optionkey = '<Table View / Stored Procedure Name>'
```

The example below (Figure 6) demonstrates querying the admin.options table to determine the version of the installed Appointments Table View (v_appointment).

Figure 6



SQL Query Text:

```
SELECT optionkey, optionvalue FROM admin.options WHERE
optionkey = 'v_appointment'
```

	optionkey	optionvalue
▶	v_appointment	2
*		

Exception Handling

See ‘Dentrix API – Exceptions Reference.pdf’ for details on exception handling for the Dentrix API.

Technical Support

Please submit technical questions or issues relating to the Dentrix API to our DDP Support team by logging in at www.dentrix.com/ddp, browsing to the Resource Center page and clicking the "Contact DDP Support" link.

For technical assistance with the Dentrix software, please contact Dentrix Software Support at 800-336-8749.